CONSTRAINED PDE OPTIMIZATION METHODS FOR MOTION SEGMENTATION AND LAYER EXTRACTION

A Dissertation
Presented to
The Academic Faculty

By

Fareed Jafri

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2018

CONSTRAINED PDE OPTIMIZATION METHODS FOR MOTION SEGMENTATION AND LAYER EXTRACTION

Approved by:

Dr. Anthony J. Yezzi, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sung H. Kang
School of Mathematics
Georgia Institute of Technology

Dr. Erik Verriest
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 09 26 2018

We are what we repeatedly do. Excellence then, is not an act, but a habit.

Aristotle

Dedicated to my parents and grandparents, specially my grandfather, M. H. Jafri.

ACKNOWLEDGMENTS

I gratefully acknowledge the help, support and guidance of my advisor Dr. Anthony Yezzi which has been phenomenal in helping me stand where I am today. I am appreciative of the collaboration with Dr. Martin Mueller as a part of this research. I would also like to acknowledge and thank all the respected professors at Georgia Tech who have mentored and educated me, and people at the academic office, specially Ms. Jacqueline Trappier, whose help and support has made my stay at Georgia Tech very memorable. I am truly grateful to the Fulbright program and the Institute of International Education (IIE) for sponsoring my education in the United States. Last but not least I am sincerely thankful to all my family members, specially my mother Dr. Abida Zaeem Jafri, my brother Aziz and my spouse Azka who have been very caring throughout my academic endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A layered representation of images allows us to capture motion, shape, appearance and occlusion structure without going into the complexity of a full 3D representation of the scene. A unified computational framework which integrates much of the current and prior work on layered models, would aid our understanding and development of layer extraction algorithms. The objective of the proposed research is to present a unified variational framework for building generative layered models that have flexibility in modeling shape, appearance, motion and occlusion structure for objects in a set of images.

This problem is approached in the framework of PDEs and calculus of variations. More specifically the strategy is developed using active contours due to their ability to capture shape and topology in an arbitrarily flexible way. The proposed approach builds on the Mumford-Shah style segmentation [2] by adapting it to a layered framework instead. A novelty of this modeling technique is that it relaxes the *brightness constancy constraint* for moving objects making the model a better fit for tracking objects in most real life scenarios. The technique links to the simplification of an earlier approach to layering [1] that used diffeomorphic maps instead of active contours. This simplification significantly reduces the computational complexity of the model yet still provides enough modeling richness for pertinent applications. More importantly a subtle yet fundamental modeling flaw in this earlier work is discovered which accidentally penalized layer occlusion as images with a more heterogeneous appearance were used (which is where the model should have excelled). The cause of this is traced to a bias in the original formulation [1] that unintentionally penalized layer occlusion thereby producing the effect. The problem is resolved by replacing the prior joint shape and appearance optimization strategy with an alternative Lagrangian style constrained shape optimization subject to PDE based appearance constraints. Bringing out the true potential of the proposed technique, MOVE (Moving Object Video Encoding), a novel technique for representing video is introduced.

CHAPTER 1
INTRODUCTION

1.1 Motivation

Layered models are commonly used in computer vision to estimate the shape, appearance, depth ordering, occlusion structure and motion of objects from a set of images, offering computationally simpler alternatives to full 3D scene models. A unified computational framework for the various modeling elements (shape, appearance, motion and depth ordering), which integrates much of the current and prior work on layered models can give us a better understanding and grip on layer extraction algorithms. A notable earlier work by Jackson et al.[1] sought to provide such a framework in the context of variational methods, elegantly cast as a single joint optimization problem. However, it did not perform as anticipated and has not been further developed. As the complexity of their formulation may have hindered its continued exploration, their diffeomorphic approach is reformulated within the much simpler framework of active contours. More importantly, though, a subtle yet fundamental modeling flaw in this earlier work which accidentally penalized layer occlusion and severely degraded the layered structure estimated by their technique is discovered. The problem associates with their improper adaptation of the classical Mumford-Shah segmentation paradigm (where occlusion does not occur) to the extraction of layered structures (where occlusion is a fundamental new dimension to the problem). The problem is fixed by presenting a new formulation that follows an augmented Lagrangian style constrained optimization process using PDE based constraints for the layer appearances while optimizing the layer boundaries rather than their suggested joint appearance and boundary optimization. This new approach better adapts the classical Mumford-Shah model from simple segmentation to occluding (and moving) layers that fixes the unnoticed problem associated

with occlusion and yields far superior results. A key novelty of this approach is that the *brightness constancy constraint* (constraint on moving pixels to have the same intensity as they move) has been relaxed which makes the technique more suitable to capturing moving objects in real life scenes. The technique also finds applications in detecting the motion of objects having a noisy appearance and video compression.

1.2 Literature Survey

1.2.1 Prior Work

Layered models are a popular and useful way to explain a set of images by a set of moving and overlapping planar regions (ordered depth-wise) that capture both the shape and appearance of projected scene objects using image matching combined with shape and/or appearance priors. Compared to a mere segmentation of an image they allow us to estimate not only the shape and appearance of objects but also their depth ordering, occlusion structure and motion without estimating a (computationally expensive) full 3D representation of the scene. Layered models on their own are useful for both appearance and motion based segmentation. Several authors [4, 5, 6, 7, 8, 9, 10] utilize the framework of layered models for segmenting images into regions of homogeneous motion and/or appearance. They can also be used to aid the 3D reconstruction of a scene [11, 12, 13]. Correspondingly several authors have used layered models for stereo reconstruction where a set of binocular images can be registered via layered models [14, 12, 15]. Another application of layered models is motion tracking with layers being used to differentiate tracked objects from untracked ones [16, 17, 18].

The problem of layer extraction from an unlabeled set of input images has been addressed in several ways. Several authors [4, 6, 7, 11, 17] use an optical flow based approach using flow fields to extract regions of homogeneous motion as layers. Alternatively authors have followed a classification approach in a Bayesian framework for extracting layer information from a set of images [19, 20, 17, 21, 22, 23, 24, 18]. Kumar et al.[25] suggested

an unsupervised learning approach using loopy belief propagation to create a generative model for the layers. A method of estimating motion and depth ordering for layers by edge tracking was proposed by Smith et al.[19] whereas Sun and Liu [26] took an approach that modeled the layers locally and jointly estimated motion and occlusion structure. Several authors [19, 27, 28] used a maximum likelihood estimate (MLE) and/or minimum description length (MDL) approach for creating a layered structure to represent a set of images. Techniques using Markov Random Fields and/or graph cuts that minimize a discreet cost function for building the layers have also been very popular [5, 25, 20, 14, 29, 22, 15, 30, 31].

Techniques targeting specific attributes of layered models have been developed as well. Works such as [19, 5, 6, 26, 7, 11] focus on an accurate motion description whereas [4, 5, 8, 9, 10] focus on improved region segmentation. Techniques for better estimation of depth ordering and occlusion structure can be found in the works of [19, 26, 7, 32, 17, 8, 33, 27, 15]. Several authors [21, 8, 34] have also suggested methods to reduce complexity and/or computational cost. Authors have also built layered models with improved resolution and/or appearance [22, 12, 30] or having better capabilities of handling large motion in the layers [4, 31] and superior motion tracking [17, 18].

1.2.2 Generative Layered Models: Potential and Challenges

Layered models are a powerful tool in computer vision that have been used for various applications. Compared to a mere segmentation of a single image they allow us to estimate the depth ordering, occlusion structure and motion of objects within a set of images in addition to their shape and appearance without going into a computationally expensive full 3D representation of the scene. Fundamentally they interpret a set of images as a set of overlapping planes and a set of motion models. Some layered models are rich enough to reconstruct the set of images that they model with acceptable quality. These are known as generative layered models. They are resourceful, and can been utilized for various pur-

poses.

Generative layered models are a richer class of layered models encapsulating a lot of what layered models in general can accomplish. Layered models can be independently used for doing an appearance based segmentation or a motion based segmentation [4, 5, 6, 7, 8, 9, 10]. They can also be used an intermediate step towards the 3D reconstruction of a scene [11, 12, 13]. A closely related problem of stereo reconstruction has also been addressed in the context of layered models [14, 12, 15]. Another popular application of layered models has been motion tracking [16, 17, 18]. Correspondingly a generative layered model that has the fullest generality in modeling the necessary elements can be useful in providing the functionality needed for such applications.

Building a fully generic layered model that has flexibility in modeling shape, appearance, motion, depth ordering (and occlusion structure) is a challenging problem. This is simply due to the fact that a fully generic model entails a lot of (unknown) information that needs to be accurately determined. Algorithms typically begin to loose their robustness in the estimation of one or more of these quantities as the information that needs to be determined increases. A vast category of algorithms that are based on minimization techniques are predisposed to getting stuck in local minima as the number of unknowns becomes larger. Therefore typically some kind of regularity or prior on the shape, appearance, motion or even occlusion structure (of the layers) is used to aid the algorithm. This however in turn leads to the problem of the algorithm loosing accuracy in the estimation of these quantities, where the loss typically depends on how much prior information is used. Therefore building a layered model that is generic and also capable of yielding accurate results is a challenging problem.

1.2.3 Prior Unified Variational Framework for Generative Layered Models

Several years ago a variational approach towards building a generative layered model was proposed by Jackson et. al [1]. They integrated the inference problem of all the model-

ing elements (shape, motion, appearance and occlusion structure) into a single variational framework. The theoretical advantage of their approach is that they cast the inference problem of all layering elements into one *single* joint optimization problem where all of the modeling elements are estimated simultaneously, starting from a completely unknown state. A resulting practical advantage was that it [1] could be inserted into a vision system to refine the outputs (shape, motion and texture) of many of the previously discussed methods (Section. 1.2.1) to make the output less dependent on the particular choice of method used.

To elucidate the theoretical advantage of their approach, the inference problem of all layering elements was cast into a single joint optimization problem where all the elements were estimated on equal footing without any prior knowledge[1] and without biasing the outcome of one based on the estimation of the other. It had the fullest generality in modeling shape, appearance, motion and occlusion structure. Given its integrated nature and versatility, their model should have performed at par with many of the other techniques mentioned in Section. 1.2.1. However, surprisingly the model showed an unexplained degradation in performance on many data sets where it was designed to excel. Despite the modeling richness that it offered, it did not perform as anticipated and was not further advanced since its introduction in 2008. Had this potentially powerful technique performed as expected it is likely that it would have captured and explained the behavior in much of the work done in the context of layering by others.

1.2.4 The Contribution

A unified variational framework for layered models that can integrate and explain much of the work done in layering is presented. Techniques to further improve the performance of the model as well as potential applications for it are also discussed. In particular the proposed strategy follows the line of work of [1] that did not perform as anticipated and

[1] Smoothness regularizers were employed to make the problem well posed

was not developed since its proposal in 2008. A revisit of this earlier work in detail brings forward two key reasons for its lack of advancement which are addressed and fixed.

First, their diffeomorphic deformation model significantly complicated the numerical implementation and heavily raised the computational cost of their algorithm. This endowed the technique with a level of generality that exceeds what is needed in most applications, and the resulting complexity may have discouraged further development by other researchers. Second, and more importantly, a subtle modeling flaw that turned out to have serious consequences is discovered. Their seemingly natural use of Mumford-Shah style appearance models [2] did not behave as expected when extended to layering and caused an unwanted shrinking effect on occluding (foreground) layers.

To obtain a more computationally efficient algorithm their model is reformulated by replacing their diffeomorphic maps with level set based active contours to capture layer shapes coupled with an affine motion model that allows for rotation, scaling, shearing and translation of these shapes. This simplified motion model is still relevant to a wide variety of practical applications and full generality in shape modeling is still maintained. Active contours are indeed known for their rich ability to easily model arbitrary shape and topology [35, 36, 37, 38, 39, 40, 41] and have generally been successful in both image and motion segmentation [42, 43, 44, 45, 46, 47, 48]. This simplification makes the algorithm more practical while still keeping it pertinent to a wide variety of useful applications.

To fix the theoretical flaw an unnoticed bias in the prior formulation is illustrated and fixed. The flaw indirectly penalized the amount of layer occlusion that distorted the shapes of the occluding layers. This problem is resolved by changing their unconstrained joint optimization strategy into a constrained optimization problem using augmented Lagrangian style PDE constraints. To do so the terms containing the shape and appearance regularizers taken from the classic Mumford-Shah segmentation model [2] are split into two separate energy functionals, one which is minimized subject to minimization of the other as a constraint. As such a better mathematical formulation for a unified variational approach to

layering is established and the correspondingly improved results are shown. When adapted in this way, *Mumford-Shah style modeling can be even more powerful for layering than in its original framework of segmentation*.

The implementation of the proposed technique using level set functions relates to (but is different from) the framework for (unoccluded) Mumford-Shah style segmentation [3]. It handles a more complex scenario involving moving and occluding layers instead. A novelty of this modeling technique is that the *brightness constancy constraint* (constraint on moving pixels to have the same intensity as they move) is relaxed. This is likely to be the case for most (moving) objects in natural scenes. This feature makes the model a better fit for capturing the motion of most objects in real life scenarios. Setting the correct mathematical foundation the concept of MOVE (Moving Object Video Encoding) is introduced that provides a novel approach towards representing video, utilizing the proposed framework.

CHAPTER 2

MUMFORD-SHAH STYLE GENERATIVE LAYERED MODELS (BI-LAYER SINGLE IMAGE)

2.1 Building a Mumford-Shah Style Generative Layered Model

Generative layered models are a powerful tool in computer vision. The are able to resynthesize a set of images by a set of planes that can move and overlap each other. Each plane typically holds specific textural details found in the images and forms a layer. These layers can also be used by the generative layered model to resynthesize any image in the set which further adds to their utility. As such they have found a number of potential applications and the problem of building them has been approached in different ways.

The proposed technique approaches the problem of building generative layered models in the context of PDEs and calculus of variations. More specifically the line of approach taken by by Jackson et al.[1] is followed. There approach to layering used Mumford-Shah [2] style appearance models and diffeomorphic mappings for modeling shape. The diffeomorphic mappings were computationally very expensive and difficult to implement. Therefore the shape modeling component is simplified by employing active contours instead of the diffeomorphic maps. This offers a huge saving in computational cost and a much more easier implementation while still offering a generous amount of flexibility in modeling shape. In doing so the proposed technique closely resembles (yet is different to) the Mumford-Shah technique [2] in image segmentation. A key motivation for this technique is that the *brightness constancy constraint* for moving pixels is relaxed which makes the model more suitable for capturing (moving) objects in most real life scenes.

2.2 Mumford-Shah Style Segmentation versus Layering

Individual image segmentation is driven by changes in appearance rather than changes in motion. The Mumford-Shah model Eq. 2.1 was proposed, and has been successfully used, for appearance driven segmentation. Its smooth appearance prior was intended to position a contour along unknown appearance boundaries. Comparatively, layer detection is predominantly driven by motion segmentation. One could, however, still leverage a Mumford-Shah style appearance prior for motion segmentation by replacing direct image-to-image intensity matching with image-to-model intensity matching. In this context a smooth appearance prior plays a fundamentally different role by relaxing the typical *brightness constancy* assumption that drives many motion segmentation algorithms. This was indeed the philosophy of Jackson et al.in [1]. Therefore, while the Mumford-Shah style appearance prior looks the same in the original formulation as well as in its adaptation to layering (Eq. 2.2), its purpose in these two contexts is very different.

Sometimes appearance boundaries and motion boundaries coincide. Figure. 2.1 shows such an example where two images could be segmented equally well by appearance alone or by motion alone. The central column of Figure. 2.1(a) shows the results of Mumford-Shah segmentation individually applied to each of the images. The central column of Figure. 2.1(b) shows its adaptation to layering in [1] jointly segmenting both the images (into a foreground and background layer). In each figure the segmenting curve is shown in red. In this case where the appearance and motion boundaries coincide the prior adaptation of Mumford-Shah to layering (Figure. 2.1(b)) works equally well when compared to its individual frame-by-frame segmentation (Figure. 2.1(a)).

On the other hand appearance and motion boundaries might not always coincide as shown in Figure. 2.2. This is likely to happen when the appearances of the (moving) objects (such as the cyclist in this example) become more heterogeneous. From the images in the central column of Figure. 2.2(a) one can already see that an (individual) appearance

based segmentation of either of the images is likely to capture several different parts of the scene as one big object. Motion based layer segmentation on the other hand is *expected* to (jointly) segment according to what can be seen in this column. In each image the desired segmenting curve is shown in red and its interior region marks the position of the foreground layer. However when the original adaptation of the Mumford-Shah model to layering as proposed in [1] is applied to this example the results shown in the central column of Figure. 2.2(b) are obtained. This happens due to an unwanted shrinking effect emanating from a modeling flaw in [1] which is elucidated and fixed in subsequent sections. Figure. 2.2(a) in reality shows the results obtained using the revised formulation in the proposed technique.

The philosophy of the original Mumford-Shah segmentation process [2] was to approximate an image with a piecewise partitioning in which the shapes and appearances of the partitions were smooth. This was cast as a joint (shape and appearance) optimization problem by minimizing an energy Eq. 2.1. The quantities to be inferred are the shape of the partitioning region R and the functions $f_{in} : x \in R \rightarrow \mathbb{R}$ and $f_{out} : x \in \Omega \backslash R \rightarrow \mathbb{R}$ representing the smooth appearances of the parts of the image interior and exterior to the region R respectively. Here Ω is the image plane, $I : x \in \Omega \rightarrow \mathbb{R}$ is the intensity function of the image, $\alpha \in [0\ 1]$ and $\beta \in \mathbb{R}$ are the weights on the appearance and shape priors respectively and ds is an infinitesimal part of the boundary of R.

Figure. 2.1(a) shows Mumford-Shah style segmentation (Eq. 2.1) applied individually to each of two frames taken from an aircraft sequence with $\alpha = 0.9$ and $\beta = 1000$. The left column in the figure shows (in red) the initial shape of R in each frame. The central column shows the final shape of R in each frame and the right column shows the appearance model for R (function f_{in} modeled over R) in each frame.

Figure. 2.1(b) correspondingly shows a layer extraction by now (jointly) segmenting the same frames using the previous adaptation of the Mumford-Shah model to layering. The left column shows the initial contour (shape of R) in each frame, the central column

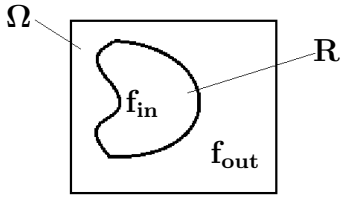
$$E(R, f_{in}, f_{out}) = \beta \underbrace{\int_{\partial R} ds}_{\text{Shape prior}}$$

$$+ \big(1-\alpha\big) \underbrace{\int_{R} \big(I - f_{in}\big)^2 dx}_{\text{Interior fidelity}} + \alpha \underbrace{\int_{R} \|\nabla f_{in}\|^2 dx}_{\text{Interior appearance prior}}$$

$$+ \big(1-\alpha\big) \underbrace{\int_{\Omega\setminus R} \big(I - f_{out}\big)^2 dx}_{\text{Exterior fidelity}} + \alpha \underbrace{\int_{\Omega\setminus R} \|\nabla f_{out}\|^2 dx}_{\text{Exterior appearance prior}}$$

(2.1)

Mumford-Shah Segmentation



Segmentation model

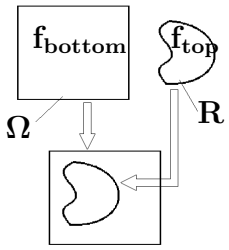
$$E(R, f_{top}, f_{bottom}) = \beta \underbrace{\int_{\partial R} ds}_{\text{Shape prior}}$$

$$+ \big(1-\alpha\big) \underbrace{\int_{R} \big(I - f_{top}\big)^2 dx}_{\text{Foreground fidelity}} + \alpha \underbrace{\int_{R} \|\nabla f_{top}\|^2 dx}_{\text{Foreground appearance prior}}$$

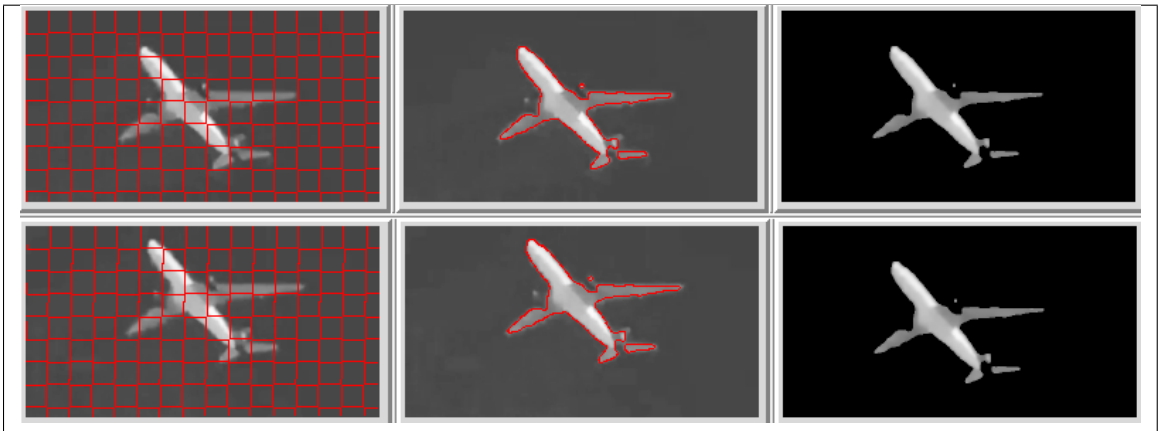
$$+ \big(1-\alpha\big) \underbrace{\int_{\Omega\setminus R} \big(I - f_{bottom}\big)^2 dx}_{\text{Background fidelity}} + \alpha \underbrace{\int_{\boldsymbol{\Omega}=\left(\mathbf{R}+\boldsymbol{\Omega}\setminus\mathbf{R}\right)} \|\nabla f_{bottom}\|^2 dx}_{\text{Background appearance prior}}$$

(2.2)

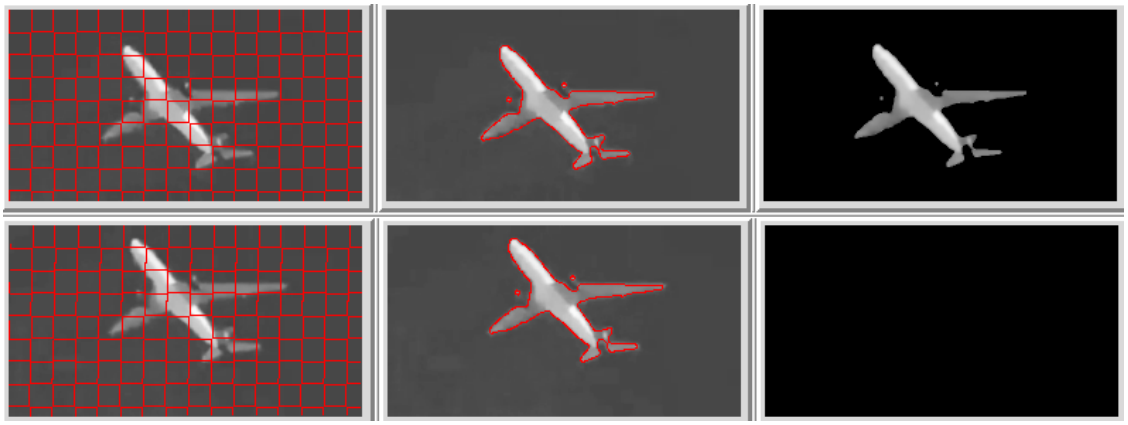
Mumford-Shah Layering



Synthesized model

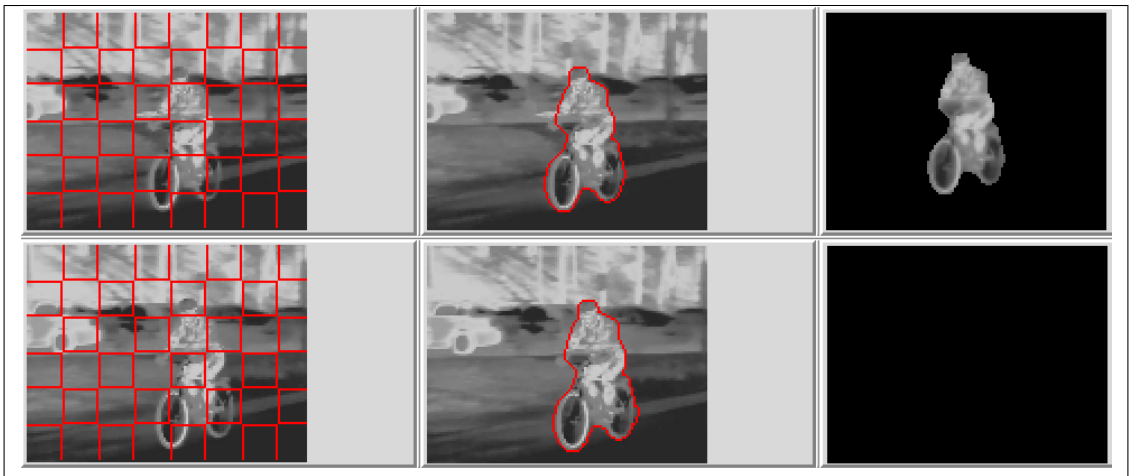


(a) Separate Mumford-Shah segmentations of two aircraft images

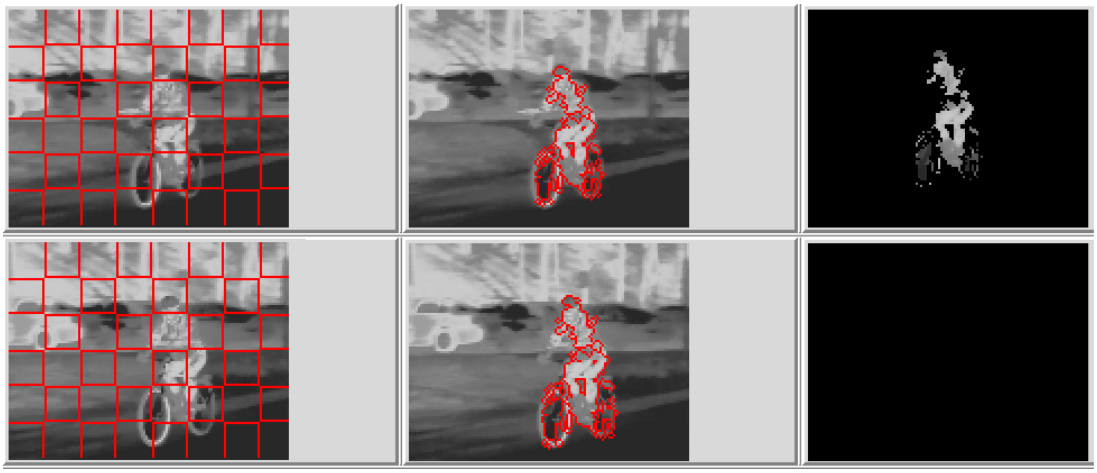


(b) Joint Mumford-Shah based layer extraction from those images yielding similar results

Figure 2.1: Mumford-Shah segmentation versus Layering: initial contour and results



(a) Expected Mumford-Shah layer extraction using two frames



(b) Actual results using the prior formulation

Figure 2.2: Revised and prior Mumford-Shah based layer extraction using two frames

shows the final contour in each frame and the right column shows the appearance model of the foreground layer (jointly extracted from the frames). Despite the similar looking results (Figure. 2.1 (a) and (b)) the two techniques are fundamentally different.

Figure. 2.2(b) (in a way similar to Figure. 2.1(b)) shows the (unexpected) results of the previous adaptation of the Mumford-Shah model to layering using two frames of a cyclist sequence instead. Figure. 2.2(a) correspondingly shows the improved results obtained through the proposed technique.

2.3 Prior (Naive) Extension of Mumford-Shah Style Segmentation to Layering

2.3.1 Layered Equivalent of Mumford-Shah Segmentation

Let us consider the adaptation of the Mumford-Shah segmentation to layering following the prior work of [1]. Eq. 2.2 shows the corresponding energy functional (for the simplest case) which generates a two-layered model using a single image. There is a foreground layer that occludes part of a background layer. Ω is the image plane (which in this example only is also the modeling region of the background layer) and $I : x \in \Omega \rightarrow \mathbb{R}$ is the intensity function of the image. R is the modeling region of the foreground layer. The quantities to be inferred are the shape (or boundary) 'C' of the foreground region 'R' and the functions $f_{top} : x \in R \rightarrow \mathbb{R}$ and $f_{bottom} : x \in \Omega \rightarrow \mathbb{R}$ representing the appearances of the foreground and background layer respectively. $\alpha \in [0\ 1]$ and $\beta \in \mathbb{R}$ are weights on the appearance and shape priors respectively and ds is an infinitesimal part of C.

The key difference between the Mumford-Shah style simple segmentation (Eq. 2.1) and its layered equivalent (Eq. 2.2) is that the former incorporates the exterior appearance prior in the exterior region ($\Omega \backslash R$) only but the latter incorporates the corresponding background appearance prior over the *entire* background domain (Ω). This is because in a layered structure the background is no longer the complement of the foreground but also extends *underneath* it. Here the background is modeled over the whole of Ω where the background appearance prior needs to correspondingly be applied. This does not gain us anything

useful for a layered model of a single image (and has only been introduced this way for a direct comparison with segmentation) but is crucial when multiple images that may occlude different parts of a layer (on account of motion) are used to *stitch together* the layer's model. However an unintentional outcome of this is that layer occlusion gets penalized. This induces a bias to shrink the regions of any occluding layers. In subsequent chapters a new formulation is presented that properly adapts the Mumford-Shah segmentation model to layered structures and removes this problem.

2.3.2 The Shrinkage Problem

When adapting Mumford-Shah segmentation to layering (and jointly optimizing appearance and shape) an unanticipated shrinking effect is observed in the regions covered by the foreground layers. This section explains it from a mathematical standpoint. Once again the simplest case (a two-layered model built from a single image) is considered for illustrating the problem.

To get a rudimentary understanding of the problem the background appearance prior in Eq. 2.2 can be decomposed giving Eq. 2.3.

$$\text{Background appearance prior} = \underbrace{\int_{\Omega \setminus R} \|\nabla f_{bottom}\|^2 dx}_{\text{Background exterior (unoccluded)}} + \underbrace{\int_{R} \|\nabla f_{bottom}\|^2 dx}_{\text{Background interior (occluded)}} \tag{2.3}$$

Substituting Eq. 2.3 into Eq. 2.2 will make it look identical to Eq. 2.1 but with an additional term labeled as 'Background interior (occluded)' coming from Eq. 2.3. This term can be minimized by simply reducing the size of the (foreground) region R. Minimization of Eq. 2.2 causes this to happen, inducing a shrinking bias on the (occluding) foreground layer and making its region smaller than expected. Also notice that this bias tends to increase as the term $\|\nabla f_{bottom}\|^2$ becomes larger, worsening the problem. This is likely to occur as

the appearance of the input images becomes more inhomogeneous and layered models built from images falling under this category are prone to suffer the most. This (shrinking) effect is seen in the foreground model (built from multiple images) in the right most column of Figure. 2.2(b).

2.3.3 A Controlled Study - Systematically Exploring the Shrinkage Problem

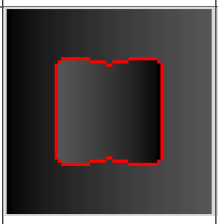
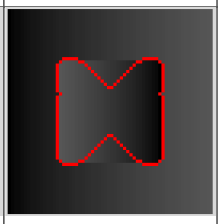
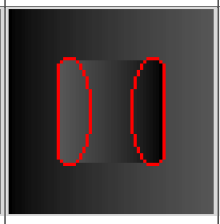
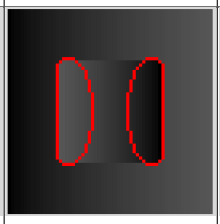
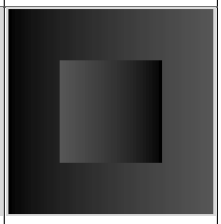
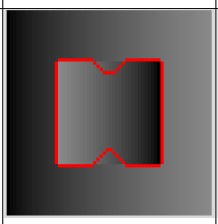
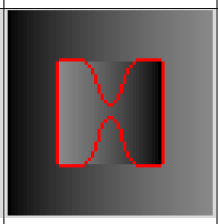
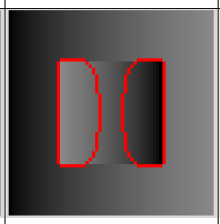
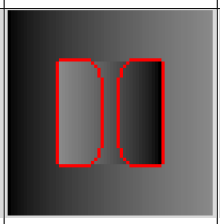
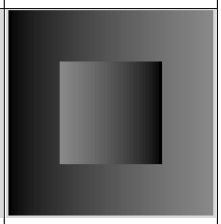
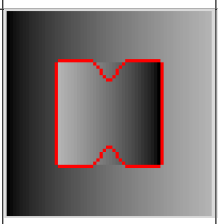
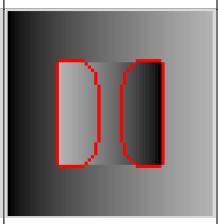
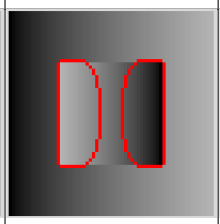
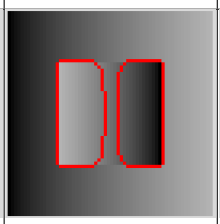
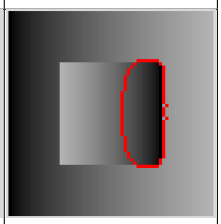
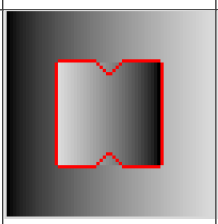
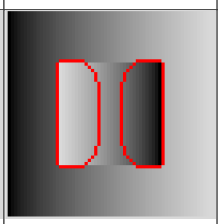
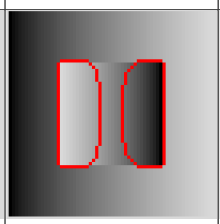
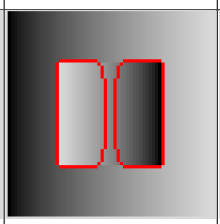
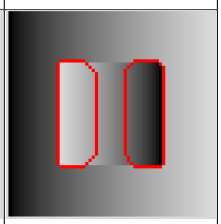
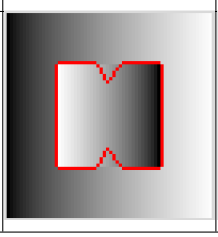
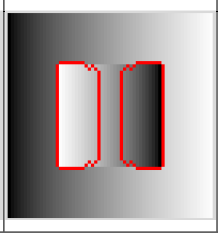
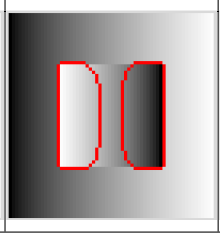
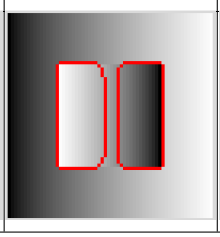
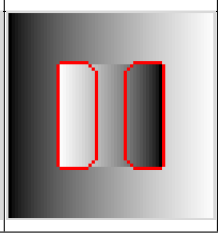
The shrinkage problem in a nutshell is about the shapes of (occluding) layers getting distorted based on (inhomogeneity in) the appearances of the layers. A systematic procedure to demonstrate the existence of the shrinkage problem is presented. A controlled experiment for (the simplest case of) a two-layered single image model is performed that shows the impact of the problem as the weight on the appearance prior (α) and degree of inhomogeneity in the appearance of the layers (or images) is changed.

To model inhomogeneity in appearance an image that has a (foreground) region of uniformly varying intensity increasing in one direction superimposed on a (background) region of uniformly varying intensity increasing in the reverse direction is taken (see Table. 2.3.3). The gradient of the uniformly varying intensities (∇I) is varied to control the degree of inhomogeneity in image appearance. The weight on the shape prior (β) is kept constant throughout the experiment (which only enforces smoothness on the shape of the curve) and has no role to play in the comparison of the results.

Table. 2.3.3 shows the results. The foreground region captured by the curve is shown for different values of ∇I and α. For values of α greater than 0.9 the weightage on the fidelity terms Eq. 2.2 is too little to produce a segmentation (unless there is a high contrast between the foreground and background regions) and for values of α less than 0.1 the the problem starts becoming ill-posed.

Ideally the contour (foreground layer) should capture the (targeted) entire central square region of uniformly varying intensity. Its failure to do so is due to the shrinking bias in the shape of the occluding foreground layer. The results show us that for a wide range values

Table 2.1: Controlled experiment: Prior (Naive) adaptation of Mumford-Shah to layering

	$\alpha = 0.10$	$\alpha = 0.33$	$\alpha = 0.66$	$\alpha = 0.90$	$\alpha = 0.99$
$\nabla I = 1.5$					
$\nabla I = 3.0$					
$\nabla I = 4.5$					
$\nabla I = 6.0$					
$\nabla I = 7.5$					

of weightage given to the appearance prior (α) and for varying degrees of inhomogeneity in image appearance (∇I), the shrinking effect is noticeable..

2.4 Solving the Shrinkage Problem

This section outlines an alternative strategy that removes the shrinkage problem discussed in previous sections. The prior optimization approach followed by [1] is now reformulated by replacing the single energy that is jointly optimized over shape and appearance with two separate energies. The shape and appearance priors that were incorporated into the single energy are now split and two separate energies (one which is appearance based and the other which is shape based) are considered that separately incorporate them. The shape based energy is optimized with the constraint that the appearance based energy remains optimized with respect to the layer appearances. Note that this is not the same as alternate optimization with respect to appearance and shape.[1] This fundamentally changes the optimization problem itself and not just merely the optimization procedure. The optimizers of the proposed method will be different from the optimizers of the prior formulation as will be demonstrated in the experimental results (Section. 3.4).

2.4.1 A Constrained Reformulation of the Problem

The proposed solution for fixing the shrinkage problem in (the simplest case of) two-layered models built from a single image is now presented. First the prior model (Eq. 2.2) is reformulated as two energies that separate the shape and appearance optimization

[1] While this may feel like an alternating optimization of shape and appearance it is not. Nowhere do the two priors (for appearance and shape) appear together.

processes as Eq. 2.4 and Eq. 2.5 respectively.

$$\begin{aligned} E_{shape} &= \beta \int_{\partial R} ds \\ &+ \big(1-\alpha\big)\left(\int_R \big(I - f_{top}\big)^2 dx + \int_{\Omega \backslash R} \big(I - f_{bottom}\big)^2 dx\right) \end{aligned} \tag{2.4}$$

$$\begin{aligned} E_{appearance} &= \alpha \int_R \|\nabla f_{top}\|^2 dx + \alpha \int_\Omega \|\nabla f_{bottom}\|^2 dx \\ &+ \big(1-\alpha\big)\left(\int_R \big(I - f_{top}\big)^2 dx + \int_{\Omega \backslash R} \big(I - f_{bottom}\big)^2 dx\right) \end{aligned} \tag{2.5}$$

The shape energy (Eq. 2.4) is now optimized with respect to the contour 'C' (boundary of R) subject to the constraint that the appearance energy (Eq. 2.5) remains optimized with respect to f_{top} and f_{bottom} for the current choice of C. The functions f^*_{top} and f^*_{bottom} are the optimized values of f_{top} and f_{bottom} in the appearance energy, Eq. 2.5 (and are not free variables to be jointly optimized with C in the optimization process of Eq. 2.4). Please see Appendix. A for a proof of the results in the remainder of this chapter. Through calculus of variations it can be shown that the functions f^*_{top} and f^*_{bottom} satisfy the constraint PDEs (Eq. 2.6 and Eq. 2.7) with vanishing Neumann boundary conditions over R and Ω respectively.

$$\alpha \Delta f^*_{top} + \big(1-\alpha\big)\big(I - f^*_{top}\big) = 0, \quad \text{over } R \tag{2.6}$$

$$\begin{aligned} &\alpha \Delta f^*_{bottom} + \big(1-\alpha\big)\big(I - f^*_{bottom}\big) = 0, \quad \text{over } \Omega \backslash R \\ &\Delta f^*_{bottom} = 0, \qquad\qquad\qquad\qquad\quad \text{over } R \end{aligned} \tag{2.7}$$

Let $\lambda_{top} : x \in R \to \mathbb{R}$ and $\lambda_{bottom} : x \in \Omega \to \mathbb{R}$ be arbitrary twice differentiable functions (acting as pointwise Lagrange multipliers) to impose the constraints (Eq. 2.6 and Eq. 2.7) into Eq. 2.4. This gives a constrained shape energy as

$$\begin{aligned} E^*_{shape} = \beta \int_{\partial R} ds \; + \int_R \lambda_{bottom} \Big(\Delta f^*_{bottom} \Big) dx \\ + \int_{\Omega \setminus R} \lambda_{bottom} \Big(\alpha \Delta f^*_{bottom} + \big(1-\alpha\big)\big(I - f^*_{bottom}\big) \Big) dx \\ + \int_R \lambda_{top} \Big(\alpha \Delta f^*_{top} + \big(1-\alpha\big)\big(I - f^*_{top}\big) \Big) dx \\ + \big(1-\alpha\big) \Big(\int_R \big(I - f_{top}\big)^2 dx + \int_{\Omega \setminus R} \big(I - f_{bottom}\big)^2 dx \Big) \end{aligned} \tag{2.8}$$

Let $\kappa : x \in C \to \mathbb{R}$ be the mean curvature of C and $N : x \in C \to \mathbb{R}^2$ be the unit outward normal to C at point $x \in C$. Calculus of variations yields Eq. 2.9 (the gradient descent PDE) for optimizing Eq. 2.8 with respect to C as well as Eq. 2.10 and Eq. 2.11 which are the PDES that λ_{top} and λ_{bottom} must satisfy with vanishing Neumann boundary conditions over R and Ω respectively.

$$\begin{aligned} \frac{\partial C}{\partial t} = -\Big(\beta\kappa - \alpha \nabla f^*_{top} \bullet \nabla \lambda_{top} \ldots \\ - \big(1-\alpha\big) \Big(\big(I - f^*_{bottom}\big)^2 + \lambda_{bottom} \big(I - f^*_{bottom}\big) \Big) \ldots \\ + \big(1-\alpha\big) \Big(\big(I - f^*_{top}\big)^2 + \lambda_{top} \big(I - f^*_{top}\big) \Big) \Big) N \end{aligned} \tag{2.9}$$

$$\alpha \Delta \lambda_{top} - \big(1-\alpha\big)\big(\lambda_{top} + 2\big(I - f^*_{top}\big)\big) = 0, \text{ over } R \tag{2.10}$$

$$\begin{aligned} &\alpha\Delta\lambda_{bottom} - \big(1-\alpha\big)\big(\lambda_{bottom} + 2\big(I - f^{*}_{bottom}\big)\big) = 0, && \text{over } \Omega\backslash R \\ &\Delta\lambda_{bottom} = 0, && \text{over } R \end{aligned} \tag{2.11}$$

To evolve the contour C first the the functions f_{top}, f_{bottom}, λ_{top} and λ_{bottom} are computed using Eq. 2.6, 2.7, 2.10 and 2.11 respectively after which C is evolved by one time step using Eq. 2.9. The process is repeated for further evolution. This fixes the shrinkage problem.

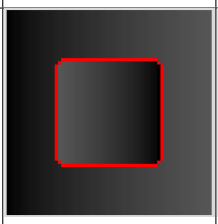
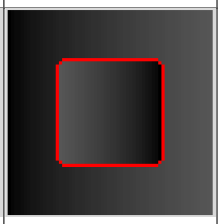
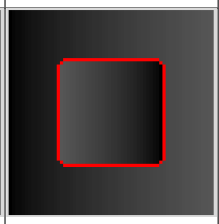
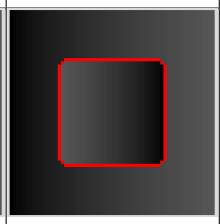
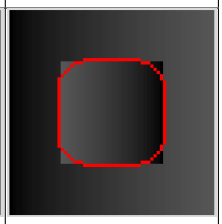
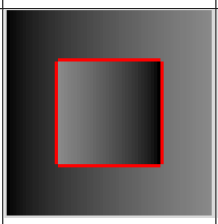
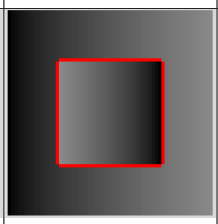
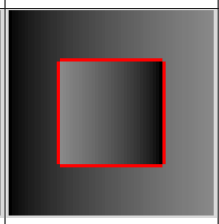
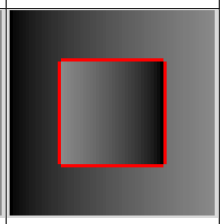
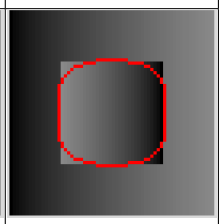
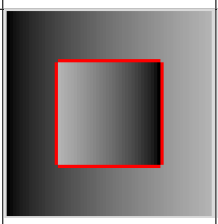
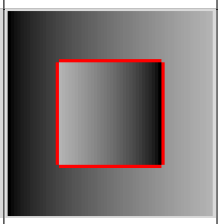
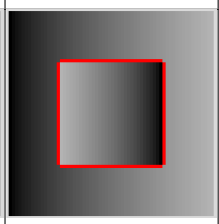
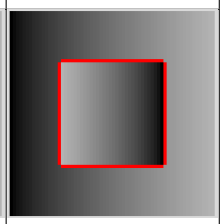
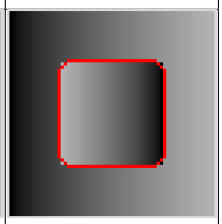
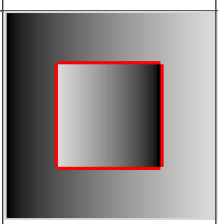
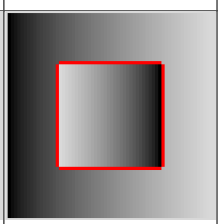
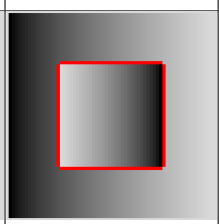
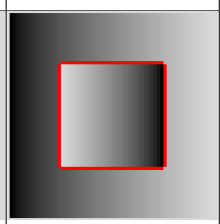
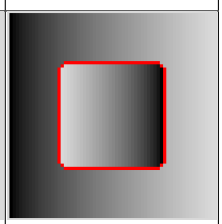
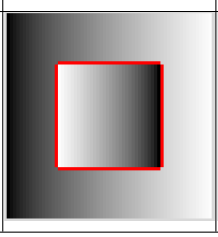
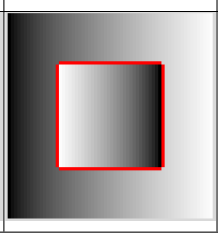
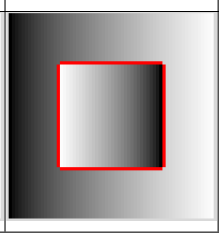
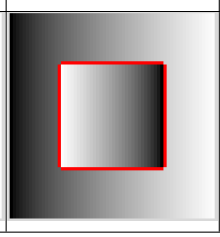
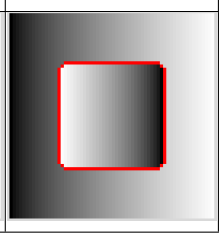
2.4.2 A Controlled Study - Improved Results with the Shrinkage Problem Fixed

Section. 2.4.1 showed the revised formulation to fix the shrinkage problem for the simplest case (a two-layered model built for a single image). The improved results (Table. 2.4.2) for the controlled experiment (Section. 2.3.3) which falls in this category are now presented using the proposed technique. Table. 2.4.2 shows the improved results that are now much closer to those that are anticipated.

2.4.3 Further Insight into the Shrinkage Problem and Corresponding Solution

The previous sections mathematically explain and visually demonstrate the shrinkage problem for the (simplest) case of a two-layered model build for a single image. For comparison purposes a mathematical explanation showing how the problem manifests itself in the curve evolution using the prior formulation is presented. Eq. 2.12 shows the gradient descent

Table 2.2: Controlled experiment: Adapting Mumford-Shah to layering with the shrinkage effect removed

	$\alpha = 0.10$	$\alpha = 0.33$	$\alpha = 0.66$	$\alpha = 0.90$	$\alpha = 0.99$
$\nabla I = 1.5$					
$\nabla I = 3.0$					
$\nabla I = 4.5$					
$\nabla I = 6.0$					
$\nabla I = 7.5$					

PDE for optimizing the original energy Eq. 2.2 with respect to the contour C.

$$\frac{\partial C}{\partial t} = \Bigg(\underbrace{\big(1-\alpha\big)\big(I - f_{top}\big)^2}_{\text{Term 1}} - \underbrace{\big(1-\alpha\big)\big(I - f_{bottom}\big)^2}_{\text{Term 2}} \cdots + \underbrace{0.5 * \alpha \|\nabla f_{top}\|^2}_{\text{Term 3}} - \underbrace{0.5 * \alpha \|\nabla f_{bottom}\|^2}_{\text{Term 4}} \cdots + \underbrace{0.5 * \alpha \big(\|\nabla f_{top}\|^2 + \|\nabla f_{bottom}\|^2\big)}_{\text{Term 5}} + \beta\kappa \Bigg) N \tag{2.12}$$

The right hand side of Eq. 2.12 displays the force (along the normal direction) on the contour. This force consists of various terms (components). A term with a positive sign will generate an inward force and a term with a negative sign will generate an outward force (along the normal direction) on the contour. Eq. 2.12 displays terms generating both inward and outward forces that involve the image data (Terms 1 and 2). There are also terms (Terms 3 and 4) generating both inward and outward forces involving $\|\nabla f_{top}\|^2$ and $\|\nabla f_{bottom}\|^2$ respectively that come from the appearance priors. There is a term involving the curvature κ whose sign (and hence direction of force generated) may vary. In addition to this there is a term (Term 5) that *always* generates an inward force involving $\|\nabla f_{top}\|^2$ and $\|\nabla f_{bottom}\|^2$. The strength of this force increases as the appearance of the image (or images) becomes more inhomogeneous. This force induces a bias in the overall force to shrink or pull the contour inwards thereby parasitically distorting its shape i.e. the periphery of the occluding region. This effect can be seen in Figure. 2.2(b) where more than one image was used to build a two-layered model.

CHAPTER 3

THE MULTILAYERED MODEL - STITCHING MULTIPLE MOVING LAYERS FROM MULTIPLE IMAGES

3.1 Generalizing the Constrained Reformulation - Multiple Images and Moving Layers

In this chapter the generalized multilayered model is presented. An extension of the reformulation to multiple layers extracted from multiple images that incorporate motion is shown.

3.1.1 Symbols and Notation

Figure. 3.1 shows the generalized multilayered model. There are L layers and N images. There is one background layer (layer 1) and there are $L-1$ foreground layers where layer k is allowed to occlude layer l $\forall k > l$. For layer l $R_l \subset \mathbb{R}^2$ is the modeled region $f_l : x \in R_l \rightarrow \mathbb{R}$ is the appearance function, α and β_l are weights on the appearance and shape priors respectively, $C_l \subset R_l$ is the boundary of R_l, ds_l is an infinitesimal part of C_l, $\kappa_l : x \in C_l \rightarrow \mathbb{R}$ is the mean curvature of C_l at x, $\gamma_n = \frac{1}{N}$ is the weightage given to image n, $\Omega_n \subset \mathbb{R}^2$ is the domain of image n, $I_n : x \in \Omega_n \rightarrow \mathbb{R}$ is the intensity function for image n and $g_{l,n} : x \in R_l \rightarrow y \in \Omega_n$ is a parameterized group action containing K parameters $p_{l,n,k},\ k \in [1\ K]$ (to represent a motion model) which maps every point $x \in R_l$ into a corresponding point in Ω_n. The functions $visible_n : x \in \Omega_n \rightarrow \mathbb{Z}^+$ and $next_n : x \in \Omega_n \rightarrow \mathbb{Z}^+$ are visibility functions that give the visible (or topmost) layer and the next (visible) layer respectively at a point x in Ω_n. These functions also influence the segmentation process of a particular layer based on the layers that occlude it (Eq. 3.5). In Figure. 3.1 at the sample point x we have $visible_n(x) = l+1$ and $next_n(x) = l-1$.

3.1.2 The Mathematical Reformulation

Corresponding to Eq. 2.4 and Eq. 2.5 the shape and appearance optimization processes are separated for the multilayered model as Eq. 3.1 and Eq. 3.2 respectively.

$$\begin{aligned} E_{shape} &= \sum_{l=1}^{L} \beta_l \int_{\partial R_l} ds_l \\ &+ \sum_{n=1}^{N} \gamma_n (1-\alpha) \int_{\Omega_n} \left(I_n - f^*_{visible_n} \circ g^{-1}_{visible_n,n}\right)^2 dx \end{aligned} \tag{3.1}$$

$$\begin{aligned} E_{appearance} &= \sum_{l=1}^{L} \alpha \int_{R_l} \|\nabla f_l\|^2 dx \\ &+ \sum_{n=1}^{N} \gamma_n (1-\alpha) \int_{\Omega_n} \left(I_n - f_{visible_n} \circ g^{-1}_{visible_n,n}\right)^2 dx \end{aligned} \tag{3.2}$$

The shape energy (Eq. 3.1) is optimized with respect to the contours $C_l,\ l \in [2\ L]$ subject to the constraint that the appearance energy (Eq. 3.2) remains optimized with respect to the functions $f_l,\ l \in [1\ L]$ given a particular set of choices $C_l,\ l \in [1\ L]$. The set of functions $f_l^*\ l \in [1\ L]$ are optimizers for the appearance energy (Eq. 3.2) for a particular set of choices $C_l,\ l \in [1\ L]$. Let $\delta_{i,j}$ be the Kronecker delta function and let us define a few

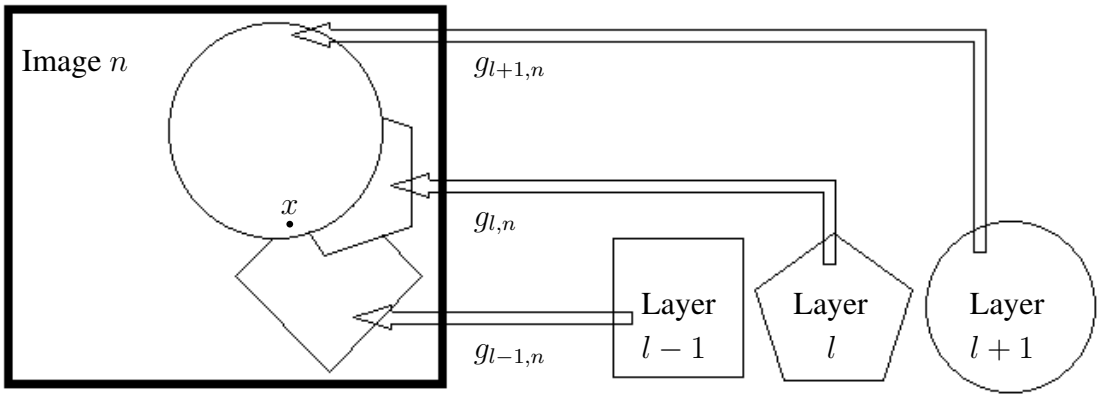


Figure 3.1: The multilayered model. A particular group action $g_{t,m}$ maps layer t to image m. x is a sample point in image n

quantities that vary pointwise for each $x \in R_l$.

$$
\begin{aligned}
&V_{l,n}(x) = (\delta_{l,visible_n \circ g_{l,n}(x)}) \det\left(\text{Jacobian}\left(g_{l,n}(x)\right)\right) \\
&\overline{\lambda}_{l,n}(x) = \lambda_{next_n \circ g_{l,n}(x)} \circ g^{-1}_{next_n \circ g_{l,n}(x),n} \circ g_{l,n}(x) \\
&D^*_{l,n}(x) = I_n \circ g_{l,n}(x) - f^*_l(x) \\
&\overline{D}^*_{l,n}(x) = I_n \circ g_{l,n}(x) - \ldots \\
&\quad f^*_{next_n \circ g_{l,n}(x)} \circ g^{-1}_{next_n \circ g_{l,n}(x),n} \circ g_{l,n}(x) \\
&G_{l,n,k} = \left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-1} \left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]
\end{aligned}
\tag{3.3}
$$

Please see Appendix. B for a proof of the results in the remainder of this chapter. Calculus of variations yields PDE based constraints (Eq. 3.4) for f^*_l with vanishing Neumann boundary conditions over the region R_l where $l \in [1\ L]$.

$$
\left(1-\alpha\right) \sum_{n=1}^{N} \left(\gamma_n V_{l,n} D^*_{l,n} \right) + \alpha \Delta f^*_l = 0 \tag{3.4}
$$

Embedding the constraints (Eq. 3.4) into the shape energy (Eq. 3.1) and using calculus

of variations yields Eq. 3.5.

$$\frac{\partial C_l}{\partial t} = -\Bigg(\big(1-\alpha\big) \sum_{n=1}^{N} \gamma_n V_{l,n} \Bigg((D^*_{l,n})^2 + \lambda_l D^*_{l,n} \dots \\ - (\overline{D}^*_{l,n})^2 - \overline{\lambda}_{l,n} \overline{D}^*_{l,n} \Bigg) + \beta_l \kappa_l - \alpha \nabla f^*_l \bullet \nabla \lambda_l \Bigg) N_l \qquad\qquad l \in [2\ L] \tag{3.5}$$

These are the gradient descent PDEs for optimizing Eq. 3.1 with respect to the contours $C_l,\ l \in [2\ L]$. Here $N_l : x \in C_l \rightarrow \mathbb{R}^2$ gives the unit outward normal to C_l at the point x and $\lambda_l : x \in R_l \rightarrow \mathbb{R},\ l \in [1\ L]$ are a set twice differentiable functions that act as pointwise Lagrange multipliers to impose the PDE constraints computed earlier (Eq. 3.4) and satisfy

$$\alpha \Delta \lambda_l - \big(1-\alpha\big) \sum_{n=1}^{N} \gamma_n V_{l,n} \big(\lambda_l + 2 D^*_{l,n} \big) = 0 \qquad\qquad l \in [1\ L] \tag{3.6}$$

with vanishing Neumann boundary conditions on R_l. To optimize Eq. 3.1 with respect to the group action parameters $p_{l,n,k},\ k \in [1\ K]$, they evolve according to Eq. 3.7 $\forall l, \forall n$.

$$\frac{\partial p_{l,n,k}}{\partial t} = -\gamma_n \big(1-\alpha\big) \int_{R_l} V_{l,n} G_{l,n,k} \dots \\ \bullet \Big(\big(2 D^*_{l,n} + \lambda_l \big) \nabla f_l - D^*_{l,n} \nabla \lambda_l \Big) dx \\ - \gamma_n \big(1-\alpha\big) \int_{C_l} V_{l,n} \Bigg((D^*_{l,n})^2 + \lambda_l D^*_{l,n} \dots \\ - (\overline{D}^*_{l,n})^2 - \overline{\lambda}_{l,n} \overline{D}^*_{l,n} \Bigg) G_{l,n,k} \bullet N_l ds_l \tag{3.7}$$

To evolve the contours first the the functions f^*_l and λ_l are computed using Eq. 3.4 and Eq. 3.6 respectively. The contours $C_l,\ l \in [2\ L]$ and the motion parameters $p_{l,n,k}\ \forall l, n, k$ are then evolved in parallel (by one time step) using Eq. 3.5 and Eq. 3.7 respectively. The

process is repeated for further evolution. Figure. 2.2(a) was obtained using the technique outlined here with $N = 2$ and $L = 2$.

3.2 The Motion Model

The layering framework presented allows for a generic motion model for capturing layer movement. Section. 1.2.4 presented an overview of the chosen motion model. More details on the model are now presented.

3.2.1 Mathematical Details

The motion model comprises a set of group actions $g_{l,n}$ $l \in [1\ L]$ and $n \in [1\ N]$ to map the layers into the images. A particular member $g_{l,n} : x \in R_l \subset \mathbb{R}^2 \rightarrow y \in \Omega_n \subset \mathbb{R}^2$ in this set allows for rotation, scaling, shearing and translation of R_l (the region of layer l) to map it into image n (Figure. 3.1). The member $g_{l,n}$ is parameterized (has six parameters $p_{l,n,k}$ $k \in [1\ 6]$) and has the form

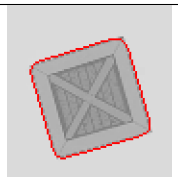
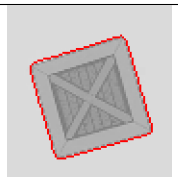
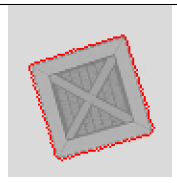
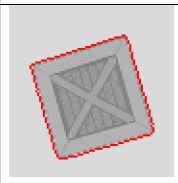
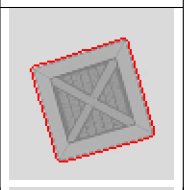
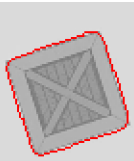
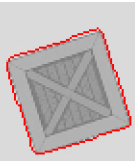
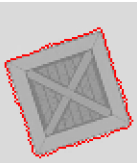
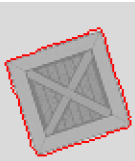
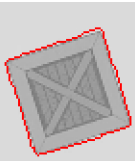
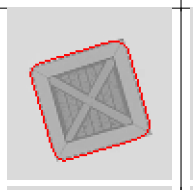
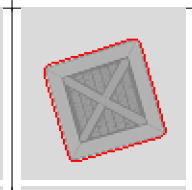
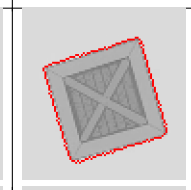
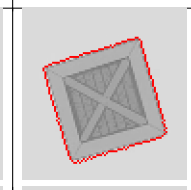
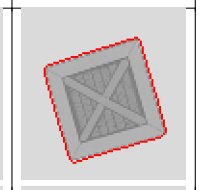
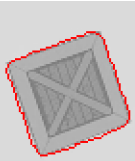
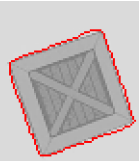
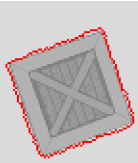
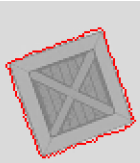
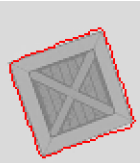
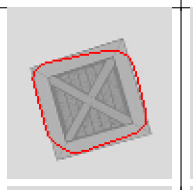
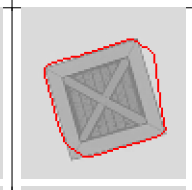
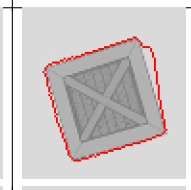
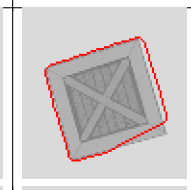
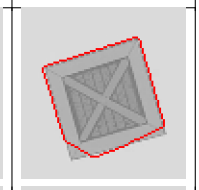
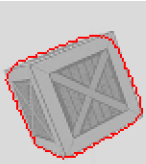
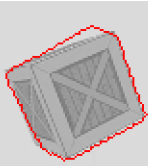
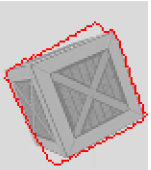
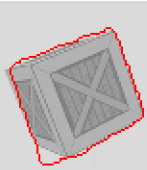
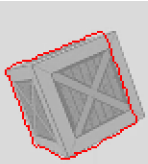
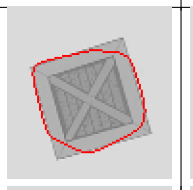
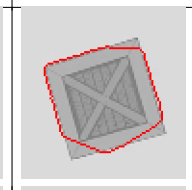
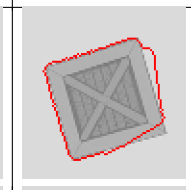
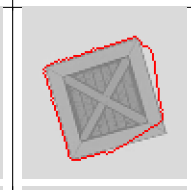
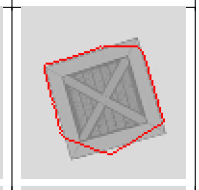
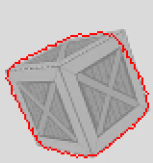
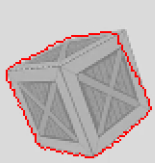
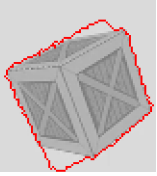
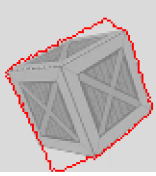
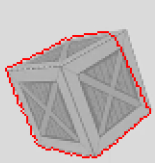
$$g_{l,n}(x) = \begin{bmatrix} \cos p_{l,n,5} & -\sin p_{l,n,5} \\ \sin p_{l,n,5} & \cos p_{l,n,5} \end{bmatrix} \begin{bmatrix} p_{l,n,3} & 0 \\ 0 & p_{l,n,4} \end{bmatrix} \begin{bmatrix} 1 & p_{l,n,6} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x - \varsigma_l \end{bmatrix} + \varsigma_l + \begin{bmatrix} p_{l,n,1} \\ p_{l,n,2} \end{bmatrix} \tag{3.8}$$

Here $x \in R_l$ is the sample point at which the motion is being evaluated and ς_l is an arbitrary point that is typically taken to be the centroid of R_l.

3.2.2 A Controlled Experiment - Limitations and Strengths of the Motion Model

A systematic procedure is now presented to show the tracking capabilities and limitations of the motion model. A controlled experiment showing how the motion model behaves when the underlying motion is not affine is presented. It is shown that in such scenarios the estimated shape (of the moving object) is likely to degrade slightly but the model still

Table 3.1: Controlled experiment: Clippings from the first and last frames showing the captured foreground layer for non-affine motion

⟵ Values of ω deg/frame ⟶		$\alpha = 0.10$	$\alpha = 0.33$	$\alpha = 0.66$	$\alpha = 0.90$	$\alpha = 0.99$
$\omega = 7.5$	Frame 1					
	Frame 2					
$\omega = 15.0$	Frame 1					
	Frame 2					
$\omega = 30.0$	Frame 1					
	Frame 2					
$\omega = 45.0$	Frame 1					
	Frame 2					

tracks the object with a fair amount of accuracy.

The experiment builds a two-layered model from two images that comprise a textured cube (treated as the foreground layer), rotating and translating over a plain background (layer). The axis of rotation and the translation vector of the cube both lie in the plain containing the two images. In each trial of the experiment the translation vector is kept constant but the (angular) speed of rotation (ω) and the weight on the appearance prior (α) are varied. A lower weight on the appearance prior increases the precision with which foreground appearance is jointly modeled from the images which in turn impacts the estimated shape. Table. 3.2 shows the shape and position of the foreground layer captured in clippings taken from the two frames for various values of ω and α. An increasing distortion in the estimated shape (contour in red) of the foreground layer (the cube) is generally observed as the speed of rotation of the target object (the cube) and accuracy of modeling appearance $(1 - \alpha)$ increases. However the curve is *still able* to reasonably track the position of the object.

3.3 Determining the Layer Count and Ordering

The previous sections discussed a generalization of the proposed model to multiple layers and images and showed how to estimate geometry, shape and motion. However, this is predicated on a known number of layers and a known order of occlusion. In some anticipated applications this will be known apriori, in particular when the proposed method is used to refine a computationally faster initial rough estimate of the layers. However even in the absence of any known apriori information a similar energy minimization principle can be used to estimate the layer count and ordering, which is outlined in the remainder of this section.

First a two-layered model with a checkered board initialization of the level set function representing the foreground layer (as shown in Figure. 2.2) is made and run to convergence. During the optimization process both occlusion orderings (by interchanging the foreground

and background models) for the layers are considered, always choosing the one that leads to the lowest energy following the gradient descent technique outlined in Section. 3.1. The converged layer yielding the highest mismatch (Eq. 3.9) is now broken into two sub layers using the same checkered board pattern (with alternate squares belonging to the original and new layer). The algorithm is rerun for the three layers once again considering both the occlusion orderings for the original and new layer always choosing the one that leads to the lowest energy following the gradient descent technique outlined in Section. 3.1. The remaining layer (that was not subdivided) is free to adapt (refine its shape, motion and appearance). Upon convergence, once again the layer with the highest mismatch is similarly split into two. The process is repeated until the mismatch score between all layers is roughly equal. At this point the algorithm has computed an optimum number of layers and their ordering.

$$Mismatch\ layer_l = \frac{\sum_{n=1}^{N} \int_{\Omega_n} V_{l,n} \left(I_n - I_n \circ g_{l,n}^{-1}\right)^2 dx}{\sum_{n=1}^{N} \int_{\Omega_n} V_{l,n} dx} \tag{3.9}$$

Figures. 3.4.2 and 3.4.2 show examples illustrating this process. In each figure the optimum layer boundaries in the initial and frames (taken from a sequence of frames) using a progressively increasing number of layers is shown. The algorithm terminates when it reaches an optimum layer count giving us the best possible output treating all the modeling elements equally. To further illustrate the underlying model that generated the layer boundaries the final part of each figure shows a resynthesis of the images in terms of the computed smooth appearance functions of the layers, which displays the relaxation of the brightness constancy constraint when capturing pixel motion. This is visible as a slight 'blurriness' throughout the images. The relaxation however is necessary as it gives the algorithm the robustness it needs to capture the layer boundaries.

3.4 Experimental Results

This section presents and compares experimental results for the proposed unified variational framework for layering using both, the previous adaptation of the Mumford-Shah segmentation model to a layered framework as well as the revised formulation. Data that includes examples from the DAVIS dataset was used. For each example the weights on the shape and appearance priors (α and β_l $\forall l$) were adjusted to give (the best results or) the least shrinkage factor (Eq. 3.10) using the prior formulation. Here $Area_{l,n}$ is the ground truth area of layer l in image n and $\widehat{Area_{l,n}}$ is the captured area by layer l in image n where N images are used to build a model of L layers.

$$\text{shrinkage factor} = \frac{\sum_{n=1}^{N}\sum_{l=1}^{L}\left(Area_{l,n} - Area_{l,n} \cap \widehat{Area_{l,n}} + |Area_{l,n} - \widehat{Area_{l,n}}|\right)}{\sum_{n=1}^{N}\sum_{l=1}^{L}\left(Area_{l,n} \cup \widehat{Area_{l,n}}\right)} \tag{3.10}$$

where $0 \leq$ shrinkage factor ≤ 1. A lower shrinkage factor indicates better performance.

3.4.1 Experimental Results

Table. 3.4 shows two-layered models built for different examples with both, the prior as well as the revised formulation using the exact same weights on the shape and appearance priors. For each example these weights were adjusted to give the best results (the least shrinkage factor as in Eq. 3.10) using the prior formulation. For each example the foreground region captured in each of the three selected frames and the corresponding model generated for the foreground layer is shown.

Table. 3.4 shows three-layered models built for different examples with both, the prior as well as the revised formulation using the exact same weights on the shape and appearance

Table 3.2: Two-layered models for different examples using the exact same parameters with the prior and revised formulation

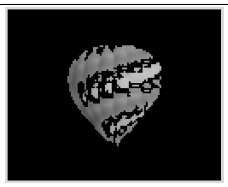
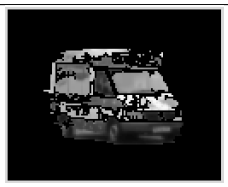
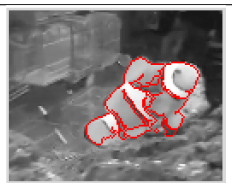
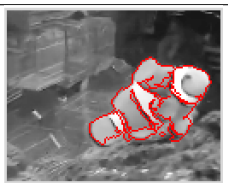
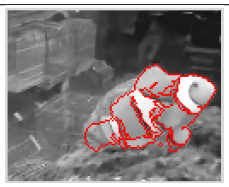
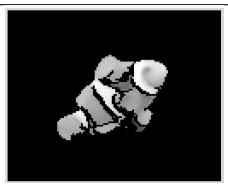
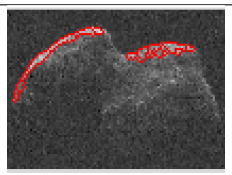
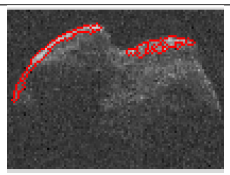
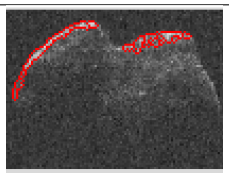
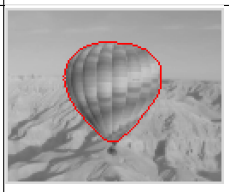
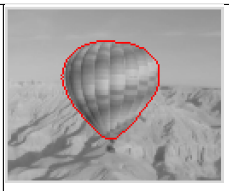
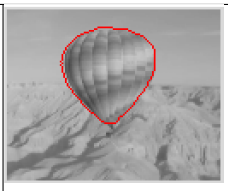
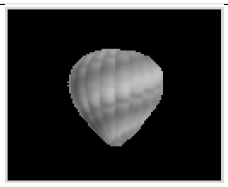
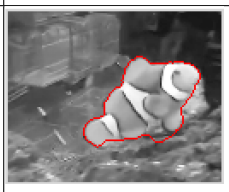
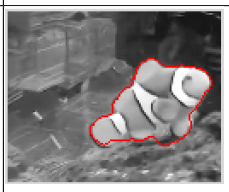
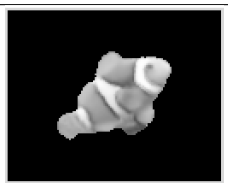
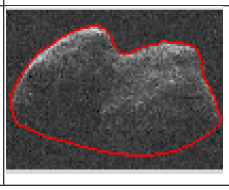
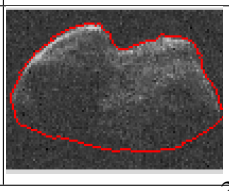
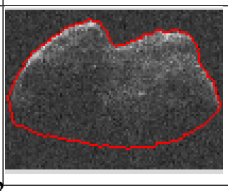
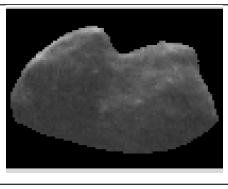
	Best Mumford-Shah style two-layered models for different examples using the prior joint optimization			
	Frame 1	Frame 2	Frame 3	Foreground Model
Hot Air Balloon (previous)				
Ice Cream Van (previous)				
Clownfish1 (previous)				
Asteroid JO-25 (previous)				
	Corresponding layered models using the exact same parameters with the revised constrained optimization			
Hot Air Balloon (New)				
Ice Cream Van (New)				
Clownfish1 (New)				
Asteroid JO-25 (New)				

Table 3.3: Three-layered models for different examples using the exact same parameters with the prior and revised formulation

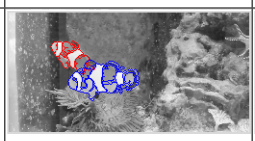
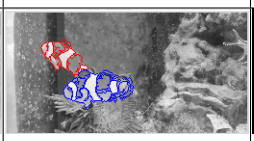
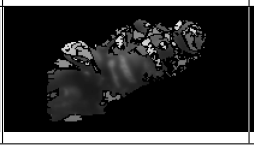
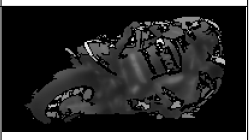
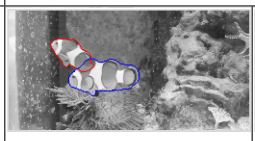
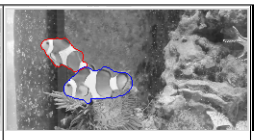
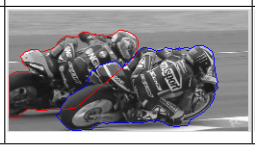
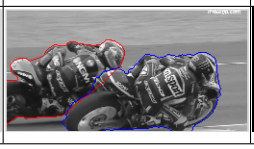
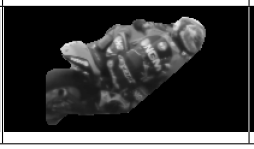
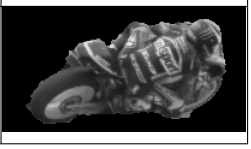
	Best Mumford-Shah style three-layered models for different examples using the prior joint optimization			
	Initial Frame	Final Frame	Layer 2 Model (red)	Layer 3 Model (blue)
Clownfish2 (previous)				
Motorbikes (previous)				
	Corresponding layered models using the exact same parameters with the revised constrained optimization			
Clownfish2 (New)				
Motorbikes (New)				

Table 3.4: Two-layered models for different examples taken from the DAVIS dataset using the exact same parameters with the prior and revised formulation

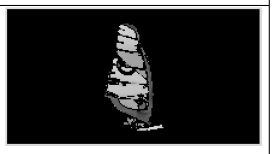
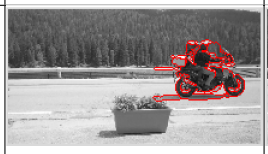
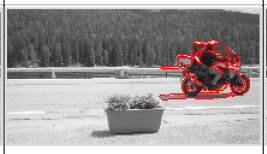
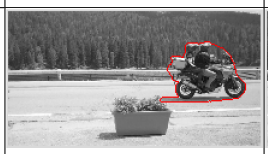
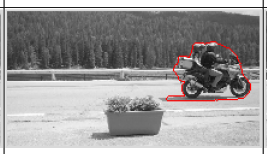
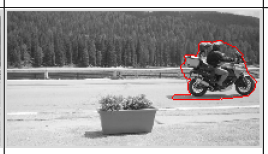
	Best Mumford-Shah style two-layered models for different examples taken from the DAVIS dataset using the prior joint optimization			
	Frame 1	Frame 2	Frame 3	Foreground Model
surf (previous)				
motorbike (previous)				
car-turn (previous)				
snowboard (previous)				
	Corresponding layered models using the exact same parameters with the revised constrained optimization			
surf (New)				
motorbike (New)				
car-turn (New)				
snowboard (New)				

Table 3.5: Three-layered models for different examples taken from the DAVIS dataset using the exact same parameters with the prior and revised formulation

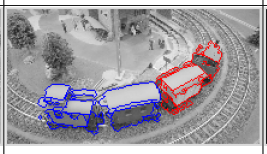
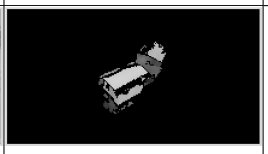
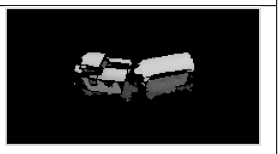
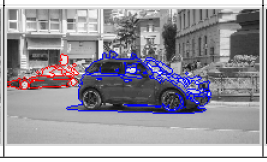
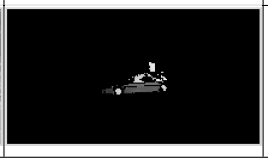
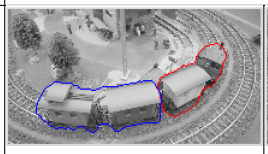
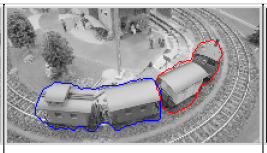
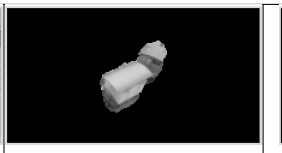
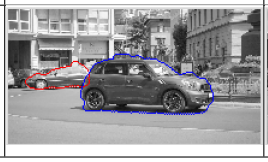
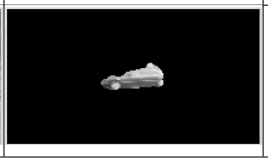
	Best Mumford-Shah style three-layered models for different examples taken from the DAVIS dataset using the prior joint optimization			
	Initial Frame	Final Frame	Layer 2 Model (red)	Layer 3 Model (blue)
train (previous)				
roundabout (previous)				
	Corresponding layered models using the exact same parameters with the revised constrained optimization			
train (New)				
roundabout (New)				

Table 3.6: The shrinkage factor for the examples shown in Tables. 3.4 to 3.4

Example	**shrinkage factor**	
	Previous	**New**
Hot Air Balloon	0.3148	0.021
Ice Cream Van	0.3016	0.009
Clownfish1	0.2178	0.015
Asteroid	0.8817	0.028
Clownfish2	0.2396	0.017
Motorbikes	0.2872	0.018
surf	0.2822	0.010
motorbike	0.1990	0.016
car-turn	0.4510	0.009
snowboard	0.0252	0.005
train	0.3165	0.015
roundabout	0.2547	0.044

priors. Once again for each example these weights were adjusted to give the least shrinkage factor using the prior formulation. For each example the two foreground regions captured in the initial and final frames and the corresponding models generated for the two foreground layers are shown. The (foreground) layer in blue can occlude the one in red. Tables 3.4 and 3.4 in a similar way show two-layered and three-layered models built for different examples taken from the DAVIS dataset. Table. 3.4 shows the shrinkage factor (Eq. 3.10) for each of the examples in Tables 3.4 through 3.4 with both, the prior as well as the revised formulation. Tables 3.4.2 and 3.4.2 revisit some of the examples shown in Tables 3.4 to 3.4 where the machine did not agree with the layer count assumed and suggested a different number of layers (using the procedure outlined in Section. 3.3) to model the scene.

3.4.2 Comments

Tables 3.4 to 3.4 show the improved results using the proposed method for different examples. Visual improvements in the results can be seen in Tables 3.4 through 3.4. The bias to penalize layer occlusion in the prior formulation which shrinks the regions of the occluding layers gets highlighted and comparatively fuller and more complete models for the foreground layers(s) using the revised formulation are observed. Furthermore the shrinkage factor Eq. 3.10 which mathematically measures the extent of this problem shows significantly improved values with the revised formulation (Table. 3.4)

A noteworthy case is the snowboard example in Table. 3.4 which shows an acceptable segmentation using both the prior and revised formulation. The high contrast between the foreground and background layers can be supported by a much simpler piecewise constant appearance model and therefore piecewise smooth appearance modeling is not needed for this example. The prior formulation (that uses the piecewise smooth appearance model) only works well in such (simpler) cases (as it did in the Aircraft example discussed in Table. 2.1). In the other examples (that truly require smooth models for appearance) the flaw in the prior formulation is exposed.

Table 3.7: Layered model of a clownfish sequence using the proposed technique. The figure shows layer boundaries in the initial and final frames using two, three and (the calculated optimum number of) four layers as well the resynthesized frames using the smooth appearance functions of the four-layered model.

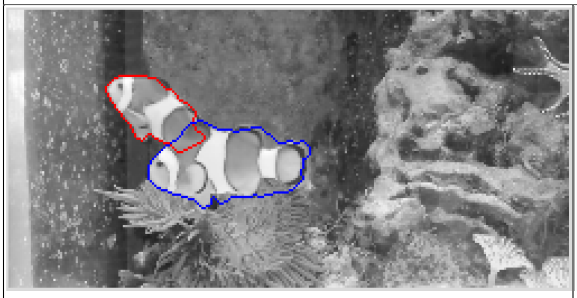
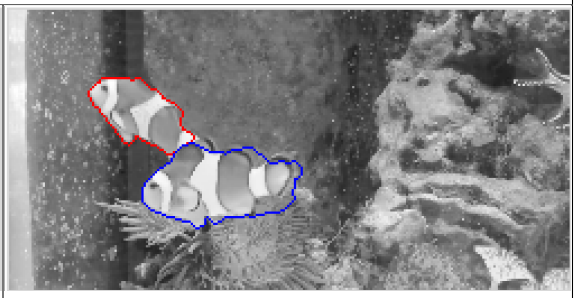
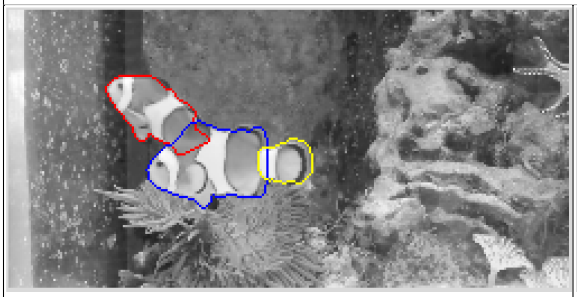
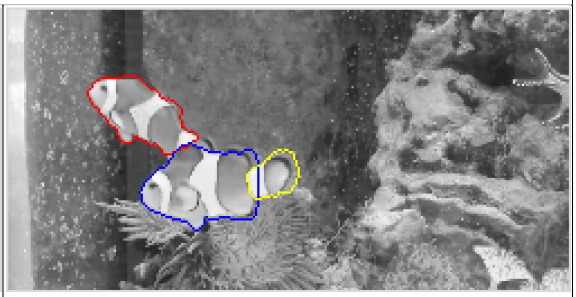
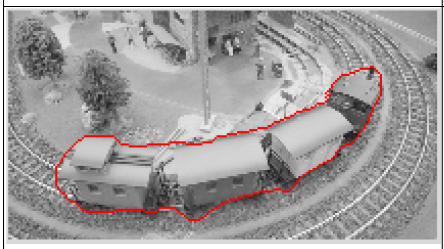
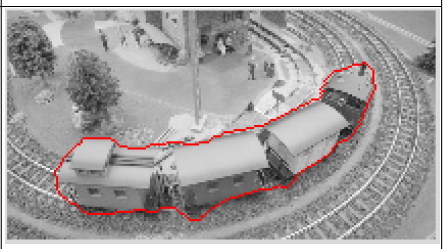
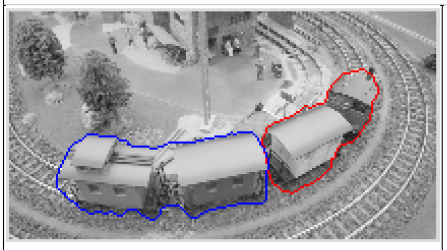
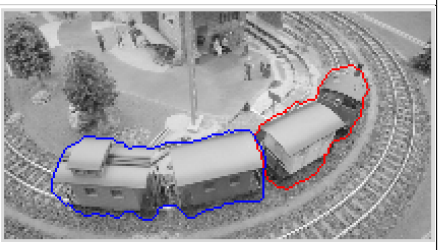
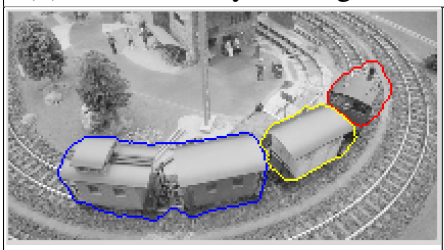
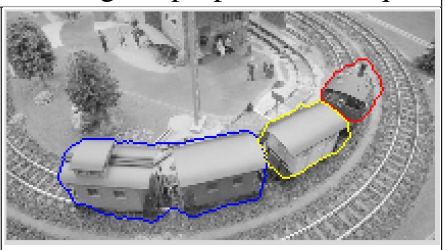
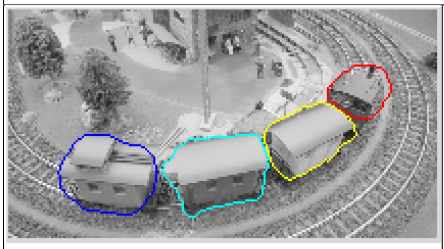
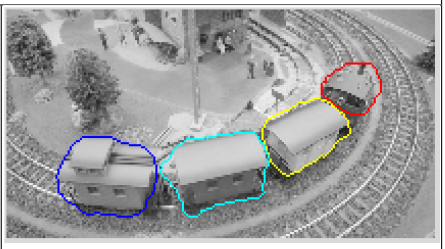
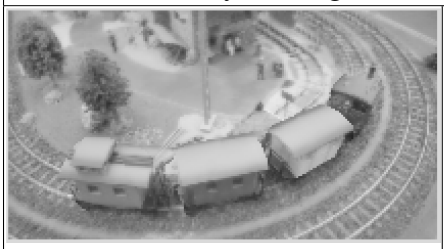
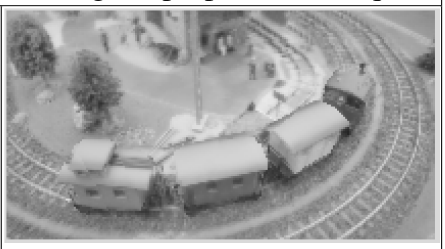
Initial Frame	Final Frame
	
(a) Best two-layered segmentation using the proposed technique	
	
(b) Best three-layered segmentation using the proposed technique	
	
(c) Best four-layered segmentation using the proposed technique	
	
(d) Reconstructed frames using the smooth appearance functions estimated in the four-layered model	

Table 3.8: Layered model of a train sequence using the proposed technique. The figure shows layer boundaries in the initial and final frames using two, three, four and (the calculated optimum number of) five layers as well the resynthesized frames using the smooth appearance functions of five-layered model.

Initial Frame	Final Frame
	
(a) Best two-layered segmentation using the proposed technique	
	
(b) Best three-layered segmentation using the proposed technique	
	
(c) Best four-layered segmentation using the proposed technique	
	
(d) Best five-layered segmentation using the proposed technique	
	
(e) Reconstructed frames using the smooth appearance functions estimated in the five-layered model	

Another noteworthy case is the train example in Table. 3.4. The example shows a scenario where there is a foreground layer (the entire train) exhibiting non affine motion. This motion can be reasonably approximated as a piecewise affine motion of more than one (non overlapping) foreground layers. The two foreground layers in Table. 3.4 are able to do this. The upside of this approach (for capturing more complex motion) is the huge saving in computational cost when compared to diffeomorphic mappings. As such this example demonstrates the sensibility in the simplification made to the model of [1].

CHAPTER 4
USING SUPERPIXELS AND TEXTURAL BLOCKS FOR CRUDE IMAGE REGISTRATION

A sensible initialization (Section. 5.3) of the algorithm can play a significant role in enhancing the performance of the proposed technique. This chapter presents a variational approach for discovering motion regions using superpixels (Section. 4.1) and textural blocks (Section. 4.2) that can then be subsequently used to initialize the algorithm. The general idea is to detect changes in motion locally by considering subregions of the image that are made based on local image intensity. An energy minimization procedure that attempts to find a set of motion parameters for each subregion is adopted. The information gathered is then processed to form motion based clusters which can then be used to initialize the main algorithm (Chapter. 3). Subsequent sections detail the process.

4.1 Registration of Superpixels

In this section a variational approach that registers maximally large superpixels to detect motion is presented. The concept, technique, its advantages and limitations are discussed.

4.1.1 Maximally Large Superpixels - Definition, Advantages and Limitations

Motion is typically observed when 'strong edges' (regions having a high intensity gradient) move. This is also true about the way the human visual system detects motion. Therefore using intensity based edges for motion tracking has been one of the adopted strategies. An example is the work of [19] that detects motion by tracking edges. However, the ultimate goal here is to find the *motion boundaries*. These are the peripheries of the independently moving objects. To do this the concept of maximally large superpixels (Figure. 4.1) is proposed. These are contiguous regions within an image having roughly the same color

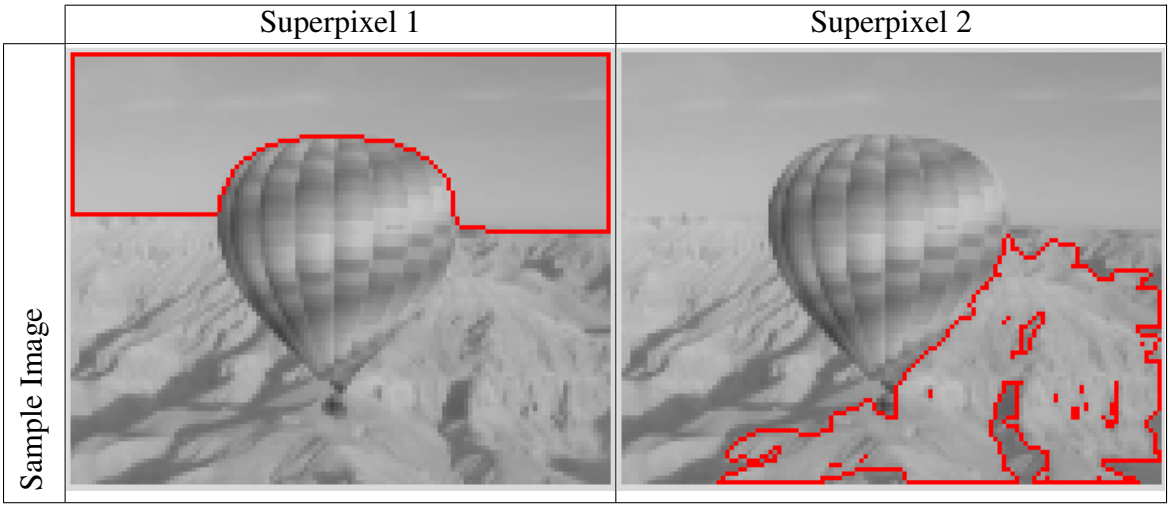


Figure 4.1: Examples of maximally large superpixels for an image taken from Table. 3.4

or intensity (or a low intensity gradient). The motion boundaries typically coincide with the boundaries of such regions which is why they are of particular interest. The following sections show how they can be leveraged to estimate motion boundaries.

To create the regions a 'region growing' algorithm is followed where a pixel is lumped with its neighbor if it has roughly the same intensity or color as that of its neighbor. Figure. 4.1 shows examples of maximally large superpixels for an image taken from one of the examples in Table. 3.4.

To infer motion boundaries between (two) images, one image acts as the reference image and the other acts as the registered image. The reference image is divided up into maximally large superpixels. The idea is to locate (register) the maximally large superpixels of the reference image in the registered image via a motion model. A variational approach to do this (Section. 4.1) is presented. The motion of the maximally large superpixels is recorded and later on further processed (Section. 4.3) to estimate motion boundaries.

It is very likely that the motion boundaries coincide with some of the boundaries of such superpixels. A maximally large superpixel whose motion significantly differs from that of its neighbors is likely to share a motion boundary with those neighbors. Hence the location of the motion boundaries can be narrowed down.

The approach however does have its limitations. The accuracy of the motion estimated for a particular superpixel begins to get compromised if the size of that superpixel becomes very small. This is because the convexity of the minimization problem used to estimate motion becomes significantly distorted for very small superpixels, leading to a number of local minimizers. This is similar to (but not exactly the same) as the well known *aperture problem*. Consequently superpixels that fall below a certain size are not registered. For most cases it was found that a sensible cut off point for making the decision to not register a superpixel was where the area of the superpixel became smaller than 5 % of area of the respective image. The motion associated with such superpixels can (usually) be more accurately detected via textural blocks (Section. 4.2.4).

4.1.2 A Variational Approach for Registering Superpixels

Details of a variational approach that can be used for registering maximally large superpixels (Section. 4.1.1) of one image in another are presented. Each superpixel is registered via a motion model. The motion model is chosen to be similar to the motion model discussed in Section. 3.2 and is a group action containing a set of (six) motion parameters that allow for rotation, scaling and translation of the superpixel. These parameters are optimized by an energy minimization principle. Eq. 4.1 shows the energy E that is minimized to estimate the motion parameters for superpixel l.

$$E = \int_{g_l\left(R_l\right)} \left(I_{reg} - I_{ref} \circ g_l^{-1}\right)^2 dx \tag{4.1}$$

Here I_{reg} is the image being registered and I_{ref} is the reference image. R_l is the region of the superpixel in the reference image and g_l is the group action containing the motion parameters to register the superpixel. The motion parameters are $p_{l,k}$ where $k \in [1, 6]$. Eq. 4.2 shows the details of the group action g_l. Here $x \in R_l$ is the sample point at which the group action is being evaluated and ς_l is an arbitrary point typically taken to be the centroid

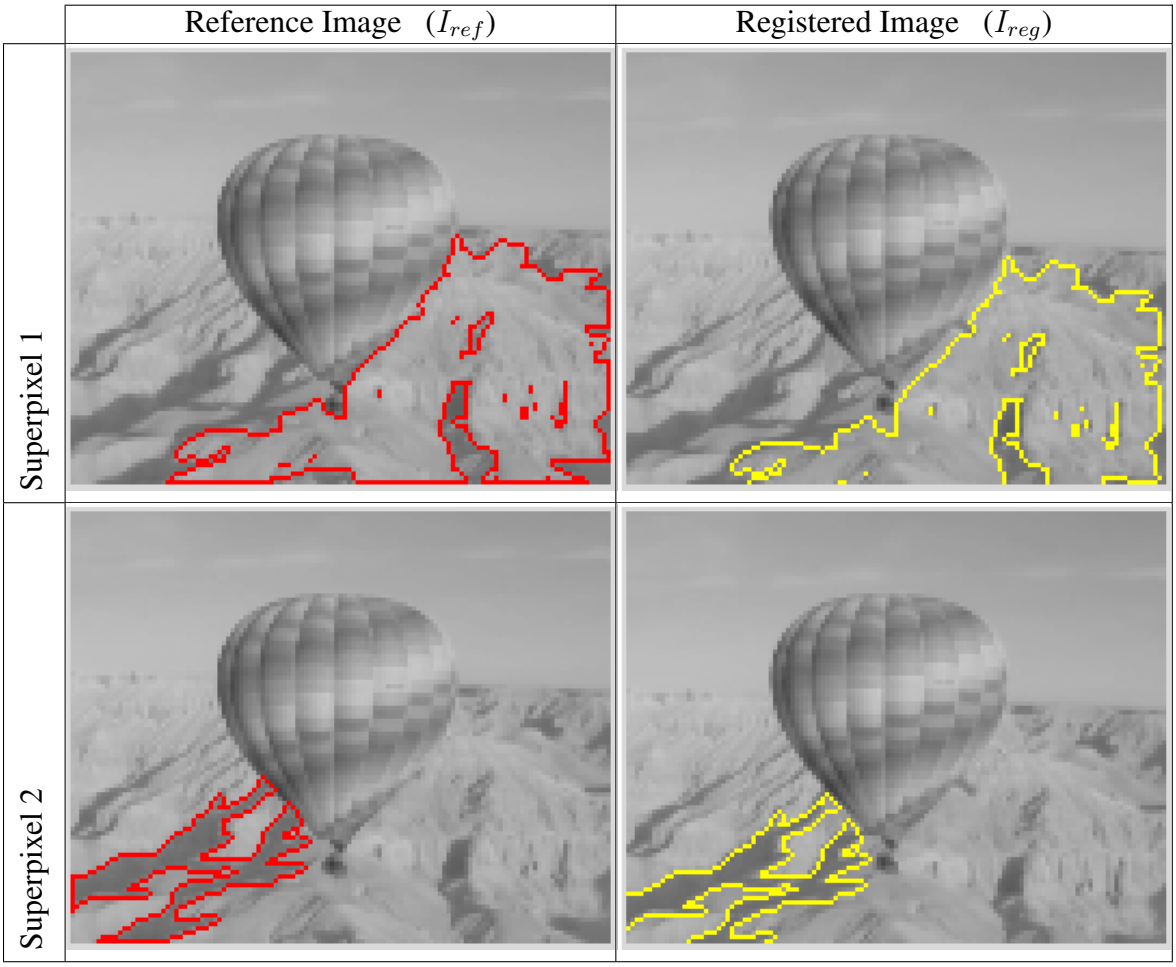


Figure 4.2: Examples of registration of maximally large superpixels for data taken from Table. 3.4. Superpixels in the reference image are shown in red and their corresponding best fit computed in the registered image is shown in yellow.

of R_l.

$$g_{l,k}(x) = \begin{bmatrix} \cos p_{l,5} & -\sin p_{l,5} \\ \sin p_{l,5} & \cos p_{l,5} \end{bmatrix} \begin{bmatrix} p_{l,3} & 0 \\ 0 & p_{l,4} \end{bmatrix} \begin{bmatrix} 1 & -p_{l,6} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x - \varsigma_l \end{bmatrix} + \varsigma_l + \begin{bmatrix} p_{l,1} \\ p_{l,2} \end{bmatrix} \tag{4.2}$$

Using calculus of variations it can be shown that a particular motion parameter $p_{l,k}$ evolves according to Eq. 4.3 to minimize Eq. 4.1.

$$\frac{\partial p_{l,k}}{\partial t} = -2 \int_{R_l} \left(I_{reg} \circ g_l - I_{ref}\right) \left[\text{Jacobian}\,(g_l)\right]^{-1} \left[\frac{\partial g_l}{\partial p_{l.k}}\right] \bullet \nabla I_{reg} dx \tag{4.3}$$

Figure. 4.2 shows registration examples of maximally large superpixels in certain images.

4.2 Registration of Textural Blocks

This section presents a variational approach that registers textural blocks to detect motion. The concept, technique, its advantages and limitations are discussed.

4.2.1 Textural Blocks - Definition, Advantages and Limitations

A textural block is a subregion of the image that shows reasonable variations in image intensity (textural details in the image). The shape of this subregion is immaterial and for all practical purposes rectangular regions can be taken. However the size is important and can have a noticeable effect on what is trying to be accomplished. Figure. 4.3 shows examples of textural blocks.

If a textural block is randomly made in one image then one can expect to find it in another image that is to be registered with the first. A motion model (set of motion parameters) can be computed to locate the textural block in the second image. A variational approach to do this (Section. 4.2.4) is presented. The motion calculated from each textural block can give us a crude estimate of the motion locally. Variations in textural block motion can then be used to make an intelligent guess about the motion boundaries. To smoothly

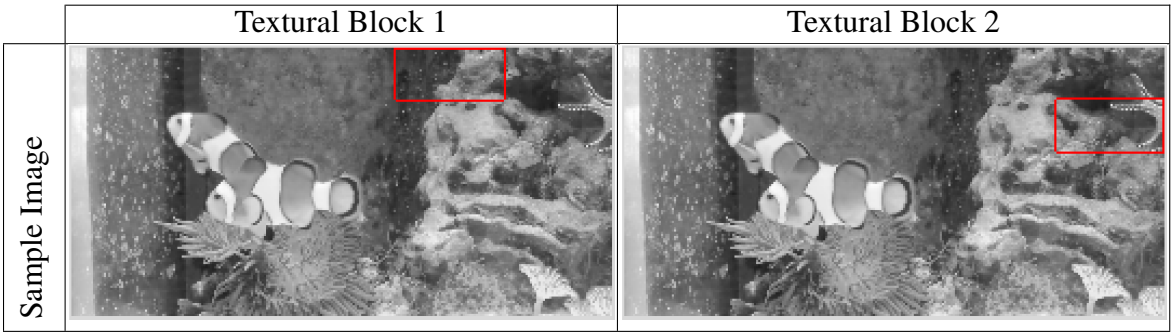


Figure 4.3: Examples of textural blocks for an image taken from Table. 3.4

track variations in motion the maximum number of unique textural blocks (of a given size) that are possible in the reference image are considered (overlap between textural blocks is allowed). The concept of textural blocks is very similar to but *not exactly the same* as that of *macro blocks* used in video compression [49, 50].

The approach does have its limitations. Firstly, blocks with flat texture (Section. 4.2.2) cannot be registered via this approach. Such blocks are completely ignored and play no role in the estimation of the motion. Secondly, the size of the block has to be at a right optimum to give correct results. Choosing a block that is too large will not allow us to capture the local variations in motion. On the other hand the accuracy of the motion estimated begins to get compromised if the size of the blocks becomes very small. This is similar to the problem that occurs when the size of the superpixels being registered becomes too small (Section. 4.1.1). Once again this is similar to but not exactly the same as the well known aperture problem. Figure. 4.4 shows textural blocks that are too large and too small to yield a reliable estimate of the motion.

4.2.2 Problem of Flat Texture

A textural block (Section. 4.2.1) that does not show enough variation in image intensity is said to have *flat texture*. Such blocks cannot be registered via the process outlined in Section. 4.2.4 because it is likely that an inconsistent motion gets estimated. This is be-

	Textural Block 1	Textural Block 2
Sample Image		
	(a) A very small textural block	(b) A very large textural block

Figure 4.4: Examples of very small and very large textural blocks shown in (a) and (b) respectively

cause the convexity of the minimization problem (in Section. 4.2.4) depends on how much intensity variation (or texture) there is in the block being registered. For a block having flat texture there could be multiple matches for it in the registered image that show up as number of local minimizers. This would translate into an incorrect local variation in the motion. Since this whole scheme fundamentally rests on detecting motion boundaries from variations in motion such estimates would distort the results and need to be omitted. Thereby blocks with flat texture are not registered and have no contribution in motion estimation. A block having a textural index (see Section. 4.2.3) less than 3 % is said to have flat texture and is not registered. Figure. 4.5 shows examples of textural blocks (taken from the same image) having flat texture.

4.2.3 The Textural Index

The textural index gives us a measure of how much texture there is in a particular textural block. For computing the textural index (τ_l) for block l the block is divided into its maximally large superpixels (Section. 4.1.1). Let R_l be the area of block l and $\Re_l^m$ be the area of the largest maximally large superpixel within this block. The textural index for block l

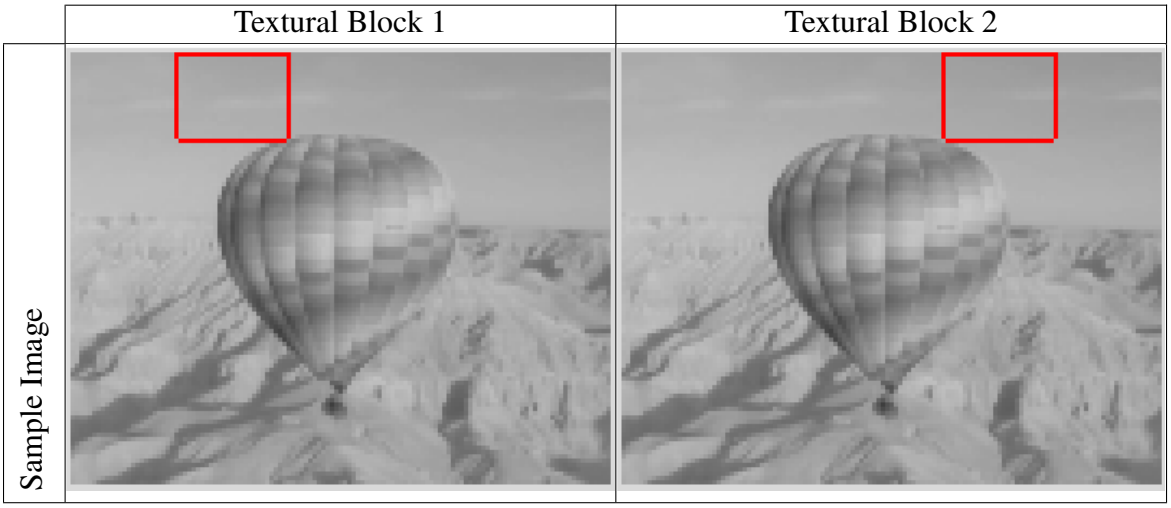


Figure 4.5: Examples of textural blocks taken from a sample image having flat texture

(τ_l) is then given by Eq. 4.4.

$$\tau_l = \frac{R_l - \Re_l^m}{R_l} \tag{4.4}$$

4.2.4 A Variational Approach to Registering Textural Blocks

Details of a variational approach that can be used for registering textural blocks (Section. 4.1.1) of one image (I_{ref}) in another (I_{reg}) are now presented. The image I_{ref} is divided into textural blocks of a reasonable size[1].The maximum number of unique textural blocks possible are considered (overlap between textural blocks is allowed). Each block in I_{ref} (that does not have flat texture) is then located in I_{reg} by estimating a motion model for the block. Once again the motion model is chosen to be a group action containing a set of (six) motion parameters that allow for rotation, scaling and translation of the textural block. A procedure identical to that outlined in Section. 4.1.2 is followed for optimizing the parameters.

The same energy Eq. 4.1 is optimized but this time to estimate the motion parameters

[1]The dimensions of the blocks are taken to be 20 % of the corresponding dimensions of the image in which they reside

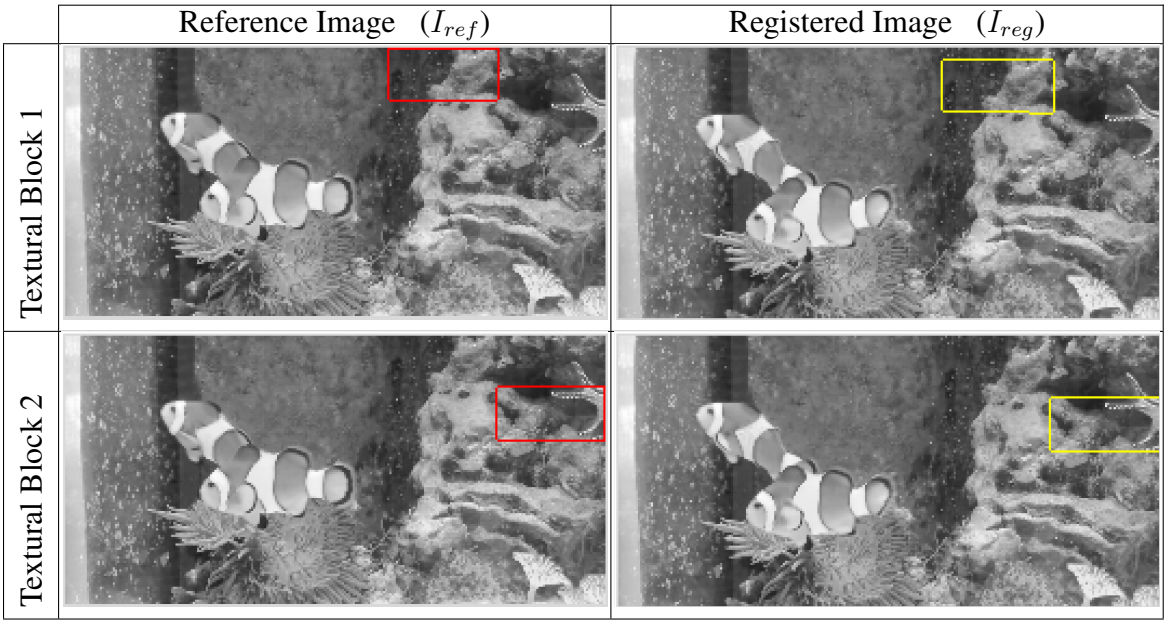


Figure 4.6: Examples of registration of textural blocks for data taken from Table. 3.4. Textural blocks in the reference image are shown in red and their corresponding best fit computed in the registered image is shown in yellow.

for textural block l. This time R_l is the region of the *textural block* under consideration in the reference image (I_{ref}) and g_l is the group action containing the motion parameters to register the block. The parameters $p_{l,k}$ where $k \in [1, 6]$ represent the motion parameters contained in g_l and are optimized using Eq. 4.3 to obtain their values for locating the block under consideration in the registered image (I_{reg}). Figure. 4.6 shows registration examples of textural blocks in certain images.

4.3 Estimating a Layered Structure from Superpixels and Textural Blocks

This section outlines a procedure to estimate independent motion regions (or layers) using a set of registered maximally large superpixels and textural blocks (Sections. 4.1 and 4.2). An initial crude estimate of the motion boundaries is made using the motion information obtained through the registration of textural blocks which is then refined using the motion information obtained through the registration of the superpixels. In the following

Table 4.1: Interpretation of the motion parameters used for registering block l

Motion Parameter	Interpretation
$p_{l,1}$	The horizontal translation of the block (in pixels)
$p_{l,2}$	The vertical translation of the block (in pixels)
$p_{l,3}$	The horizontal scale factor of the block
$p_{l,4}$	The vertical scale factor of the block
$p_{l,5}$	The angular rotation of the block (in radians)
$p_{l,6}$	The skew factor of the block

discussion textural blocks and maximally large superpixels are referred to as *blocks*.

4.3.1 Using Motion Information of Textural Blocks

Each block has an associated motion vector. This is a vector containing the motion parameters used to register the block. Let us first consider the textural blocks. Let

$$\overline{\mathbf{m}}_{\mathbf{l}} = \begin{bmatrix} p_{l,1} \\ p_{l,2} \\ \vdots \\ p_{l,k} \end{bmatrix} \tag{4.5}$$

be the vector containing the motion parameters for block l that is registered using the group action g_l. Table. 4.3.1 shows the interpretation of each motion parameter.

The goal now is to compute the motion variance vector (Eq. 4.8) at each point x in the reference image. To do this first the motion vector for each block is synchronized (see Section. 4.3.2) to the point x. Eq. 4.6 gives the synchronized motion vector $\widehat{\overline{\mathbf{m}}_{\mathbf{l}}}(x)$ when the motion vector of block l ($\overline{\mathbf{m}}_{\mathbf{l}}$) is synchronized to x.

$$\widehat{\overline{\mathbf{m}}_{\mathbf{l}}}(x) = \begin{bmatrix} \widehat{m}_{l,1}(x) \\ \widehat{m}_{l,2}(x) \\ \vdots \\ \widehat{m}_{l,K}(x) \end{bmatrix} = sync(\overline{\mathbf{m}}_{\mathbf{l}}, x) \tag{4.6}$$

The motion mean and motion variance vector at point x given by $\overline{\mu}(x)$ and $\overline{\mathbf{mv}}(x)$ respectively are now computed.

$$\overline{\mu}(x) = \begin{bmatrix} \mu_1(x) \\ \mu_2(x) \\ \vdots \\ \mu_K(x) \end{bmatrix} = \frac{\sum_{l=1}^{L} \chi_l(x) \widehat{\mathbf{m}}_\mathbf{l}(x)}{\sum_{l=1}^{L} \chi_l(x)} \tag{4.7}$$

$$\overline{\mathbf{mv}}(x) = \frac{1}{\sum_{l=1}^{L} \chi_l(x)} \begin{bmatrix} \sum_{l=1}^{L} \chi_l(x) \Big(sat_{l,1}\big(\widehat{m}_{l,1}(x) - \mu_1(x)\big) \Big)^2 \\ \sum_{l=1}^{L} \chi_l(x) \Big(sat_{l,2}\big(\widehat{m}_{l,2}(x) - \mu_2(x)\big) \Big)^2 \\ \vdots \\ \sum_{l=1}^{L} \chi_l(x) \Big(sat_{l,K}\big(\widehat{m}_{l,K}(x) - \mu_K(x)\big) \Big)^2 \end{bmatrix} \tag{4.8}$$

where

$$\chi_l(x) = \begin{cases} 1, & x \in R_l \text{ and block } l \text{ was registered} \\ 0, \text{ otherwise} \end{cases} \tag{4.9}$$

Figure. 4.7 shows the saturation function $sat_{l,k}(x)$. Table. 4.3.1 shows how to choose the values for the constant $tol_{l,k}$ for different values of l and k. Note that $\overline{\mu}(x)$ and $\overline{\mathbf{mv}}(x)$ are 0 if $\sum_{l=1}^{L} \chi_l(x)$ is 0.

The magnitude of the motion variance vector ($\|\overline{\mathbf{mv}}(\mathbf{x})\|$) gives an estimate of how close a point (x) is to a motion boundary. A point $x \in I_{ref}$ that is closer to a motion boundary will yield a higher value for $\|\overline{\mathbf{mv}}(x)\|$. To further accentuate this effect the motion indicator

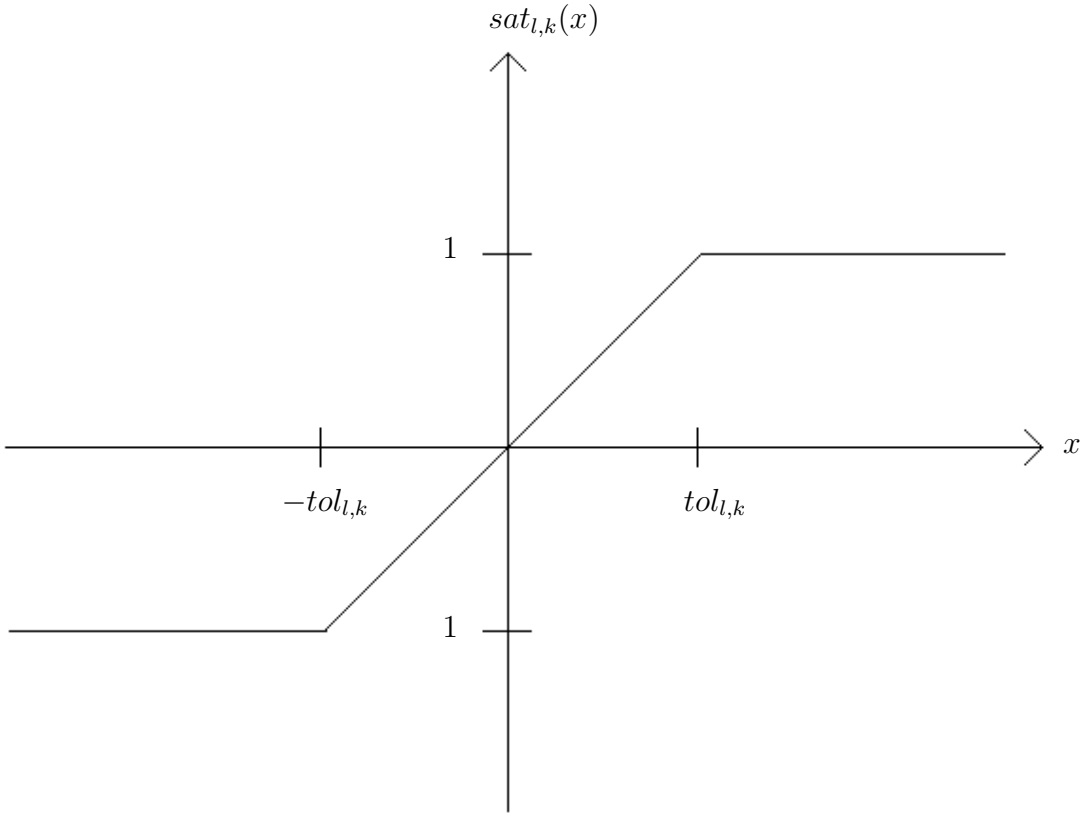
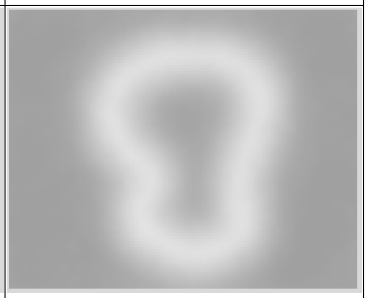
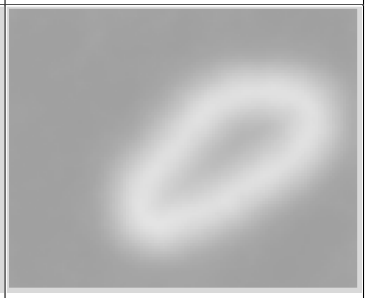
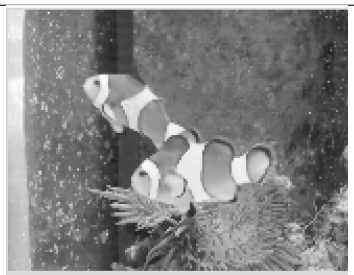
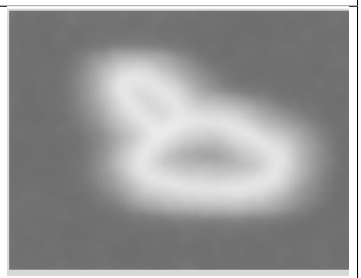


Figure 4.7: Saturation function for different values of l and k

Table 4.2: Tolerance values ($tol_{l,k}$) for different values of l and k

Tolerance	Value
$tol_{l,1}$	3 $\forall l$
$tol_{l,2}$	3 $\forall l$
$tol_{l,3}$	0.07 $\forall l$
$tol_{l,4}$	0.07 $\forall l$
$tol_{l,5}$	0.0524 $\forall l$
$tol_{l,6}$	0.2 $\forall l$

Table 4.3: Plots of the motion indicator function shown for various examples. The left column shows the reference image (I_{ref}), the central column shows the registered image (I_{reg}) and the right column shows the plot of the motion indicator function (mi) with respect to the reference image. The dark contiguous regions in mi indicate motion clusters.

	$\mathbf{I}_{ref}$	$\mathbf{I}_{reg}$	mi
Example 1			
Example 2			
Example 3			

function $mi(x)$ is considered (Eq. 4.10).

$$mi(x) = \frac{e^{\|\overline{\mathbf{mv}}(\mathbf{x})\|} - 1}{(e^{\sqrt{K}} - 1)}, \quad 0 \leq mi(x) \leq 1 \tag{4.10}$$

where K is the total number of parameters in the motion model.

Since the layer boundaries tend to typically coincide with motion boundaries this function also gives us an estimate how close a point is to a layer boundary. Table. 4.3.1 shows plots of the motion indicator function for some examples. The reference image, registered image and plot of $mi(x)$ are shown.

The plot typically partitions the region of the reference image into dark regions where each dark region can be interpreted as separate layer. Essentially these dark regions are motion clusters where the constituent pixels exhibit nearly similar motion. To further demarcate these regions only those points are considered that have a value of $mi(x) \leq 0.4$. Such points are sufficiently within a particular dark region. Using a region growing algorithm these selected points are then grouped to form motion clusters. Two neighboring points that fall within a dark region are assigned to the same cluster. Each such cluster C_i can be treated as a crude estimate of a particular layer. The motion vector $\overline{\mathbf{c}}_\mathbf{i}$ for C_i is now computed using the following formula.

$$\overline{\mathbf{c}}_\mathbf{i} = \frac{\sum_x \Gamma_i(x)\overline{\mu}(x)}{\sum_x \Gamma_i} \tag{4.11}$$

where

$$\Gamma_i(x) = \begin{cases} 1, & x \in C_i \\ 0, \text{ otherwise} \end{cases} \tag{4.12}$$

4.3.2 Synchronizing Motion to a Different Centroid

The motion vectors (for the superpixels and textural blocks which we refer to as blocks here) are computed with respect to some reference point, called the motion centroid. This is typically taken to be the centroid of the region containing the block. Let ς_l be the motion centroid for block l which was registered with the group action g_l having a motion vector $\overline{\mathbf{m}}_\mathbf{l}$. The motion (or more specifically the motion vector) can be synchronized to a new motion centroid 'y'. This entails finding an equivalent motion vector using a different reference point (y). Eq. 4.13 shows how to synchronize the motion for block l to a new motion centroid y where $\overline{\mathbf{m}}_\mathbf{l}$ is the current motion vector, ς_l is the current motion centroid and $sync(\overline{\mathbf{m}}_\mathbf{l}, y)$ gives the new motion vector synchronized to y. Note that the current

motion vector $\overline{\mathbf{m}}_\mathbf{l} = \begin{bmatrix} m_{l,1} & m_{l,2} & \dots m_{l,K} \end{bmatrix}^T$ and current motion centroid ς_l are used when evaluating the group action g_l.

$$sync(\overline{\mathbf{m}}_\mathbf{l}, y) = \begin{bmatrix} \widehat{m}_{l,1}(y) \\ \widehat{m}_{l,2}(y) \\ \vdots \\ \widehat{m}_{l,K}(y) \end{bmatrix}, \quad \text{where} \begin{bmatrix} \widehat{m}_{l,1}(y) \\ \widehat{m}_{l,2}(y) \end{bmatrix} = g_l\big(y\big) - y, \quad \widehat{m}_{l,i}(y) = m_{l,i}\ \forall i > 2 \tag{4.13}$$

4.3.3 Refining Layer Boundaries Using Motion Information of Superpixels

Once the motion clusters from the plot of the motion indicator function (outlined in Section. 4.3) are obtained they can be used in conjunction with the motion vectors of the maximally large superpixels to refine the estimates of the motion (or layer) boundaries. To do this every superpixel is assigned to a cluster. Eq. 4.5 is now used to represent the motion vector of superpixel l. The superpixels are assigned to the clusters based on the following criteria.

- **A superpixel only touches one motion cluster**

  Superpixels that touch only one motion cluster are assigned to that cluster

- **A superpixel touches multiple motion clusters and is registered**

  If a superpixel touches multiple motion clusters and was registered then its motion vector is compared with the motion vector of each cluster that it touches after synchronizing its motion vector to the centroid of that cluster. The cluster C_k with the best match is selected. To do this let superpixel l be the superpixel that needs to be assigned to a cluster. The matching score between the motion vector of superpixel l and that of cluster C_i is computed according to Eq. 4.14. Here ς_i is the motion

centroid of cluster C_i

$$Match(l,i) = \|sync(\overline{\mathbf{m}}_\mathbf{l}, \varsigma_i) - \overline{\mathbf{c}}_\mathbf{i}\| \tag{4.14}$$

The allotted cluster (C_k) to superpixel$_l$ is then given by

$$C_k \quad \text{where} \quad k = \max_i \; Match(l,i) \tag{4.15}$$

- **A superpixel touches multiple motion clusters and is not registered**

  A superpixel that touches multiple motion clusters and is not registered is assigned to a cluster using the following strategy. Let superpixel l be the superpixel that needs to be assigned to a cluster. The motion vector of each cluster that the superpixel touches is considered in turn and is used to map the superpixel from the reference image (I_{ref}) to the registered image (I_{reg}). The mean absolute distortion or MAD (Eq. 4.16) of the mapped superpixel in the registered image is computed. (Eq. 4.16) is essentially an intensity based error measure when mapping superpixel l to the registered image using the motion vector $\overline{\mathbf{c}}_\mathbf{i}$ of the cluster C_i. g_i is the corresponding group action associated with $\overline{\mathbf{c}}_\mathbf{i}$.

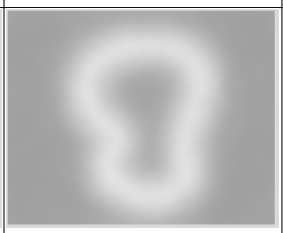
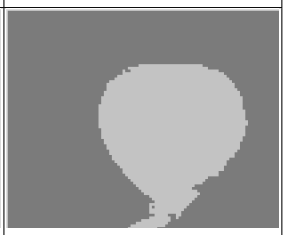
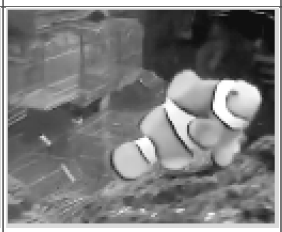
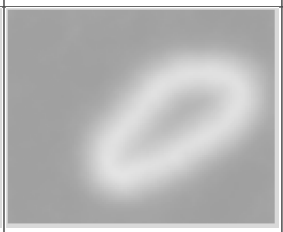
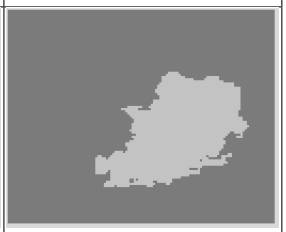
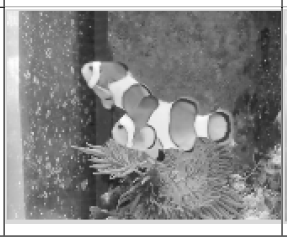
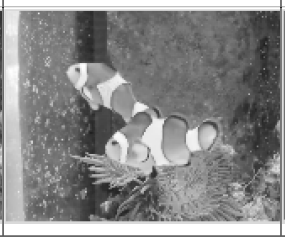
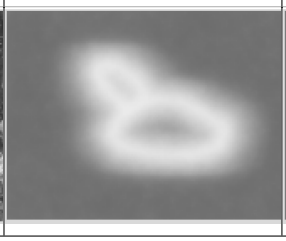
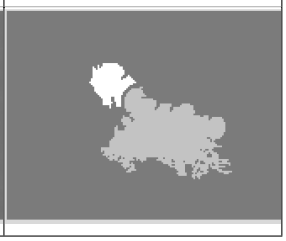
$$MAD(l,i) = \frac{\sum_{\forall x} \Upsilon(g_i(x))\chi_l(x) \parallel I_{reg} \circ g_i(x) - I_{ref}(x) \parallel}{\sum_{\forall x} \Upsilon(g_i(x))\chi_l(x)} \tag{4.16}$$

  where

$$\Upsilon(y) = \begin{cases} 1, & y \in I_{reg} \\ 0, \text{ otherwise} \end{cases} \tag{4.17}$$

  and $\chi_l(x)$ is given by Eq. 4.9. Note that $MAD(l,i) = \infty$ if $\sum_{\forall x} \Upsilon(g_i(x))\chi_l(x) = 0$. The cluster with the least MAD is assigned to the superpixel. The allotted cluster

Table 4.4: Refined motion clusters using superpixel motion for the examples shown in Table. 4.3.1. The figure shows the reference image (I_{ref}), registered image (I_{reg}), motion indicator function (mi) and final refined motion clusters.

	I_{ref}	I_{reg}	mi	Motion Clusters
Example 1				
Example 2				
Example 3				

(C_k) to superpixel l is given by Eq. 4.18.

$$C_k \quad \text{where} \quad k = \min_i \; MAD(l, i) \tag{4.18}$$

Table. 4.3.3 shows the refined motion clusters for the motion indicator plots shown in Table. 4.3.1 by using superpixel motion to assign superpixels to a particular cluster. Each refined cluster can now be taken as a separate layer in order to initialize the main algorithm (Section. 3.1).

CHAPTER 5

ENHANCING COMPUTATIONAL SPEED AND ROBUSTNESS

Generally the usage of richer models within a variational framework is likely to increase the computational time of the underlying algorithm as well as the likelihood of it getting stuck in local minimizers. Given the modeling richness of the proposed technique, these issues are consequential and need to be addressed. This chapter discusses how the computational speed and robustness towards local minimizers of the proposed method can be improved.

The issue of computational speed primarily arises because the variational methods presented entail repeatedly solving several PDEs throughout the optimization procedure, making computational speed a critical issue. In addition to this the generic nature of the model results in the corresponding minimization problem typically having a non convex topology which often leads to the algorithm getting stuck in local minimizers during the optimization process. The result may be a less than ideal representation for the layers. Therefore, methods to reduce the computational time as well as improve the robustness of the algorithm towards local minimizers are explored. A discussion on multiresolution methods, multithreading, strategic initialization and accelerated active contours is presented.

5.1 Using a Multiresolution Approach

A multiresolution approach can be useful in reducing the computational time as well as improving robustness to local minimizers. The technique finds success owing to the fact that the modeling elements (shape texture and motion) of major (moving) objects (layers) can still be inferred at a lower level of resolution. This allows for unnecessary details to be conveniently glossed over while obtaining cruder estimates of the modeling elements at the lower levels of resolution. The result is a saving in computational time as well as improved robustness towards local minimizers.

The computational burden is reduced as only necessary information is initially processed to get a rough estimate of the (unknown) modeling elements. Working at a lower level of resolution allows us to quickly get a coarse estimate of the unknowns thereby leaving the more expensive computations that augment finer details to be progressively done at higher levels of resolution. This would minimize the overall computational time in reaching the final solution when compared to an approach that works through and through at the highest level of resolution (where the computational cost would be at the maximum all the way through).

The robustness to local minimizers is improved because working at a lower level of resolution removes finer details (of motion, shape and texture etc.) which makes the topology of the minimization problem (Eq. 3.1 and 3.2) more convex. This improves the chances of the algorithm initially converging near the desired global minimum. From this vantage point the details associated with the higher levels of resolution (that bring in local minimizers many of which may have already been avoided) can now be included and processed that push the algorithm towards the global minimum. As such the multiresolution approach also improves the robustness of the algorithm towards local minimizers. In the next section results showing the improvements made from both these perspectives are presented.

5.1.1 Experimental Results - Improving Computational Speed

Experimental results showing improvements in computational speed using a multiresolution version of the algorithm are presented here. Each experiment starts with a partially informed guess. As such an initialization that knows both the layer count and ordering apriori is used. A separate layer is allowed to model each of the independently moving objects whose count and depth ordering is known. The layer boundaries are initialized to bounding boxes that surround their respective target objects. The ordering of each layer is dictated by the (known) depth ordering of the object it surrounds. The other modeling elements (such as shape and texture) are completely unknown. For this same initialization

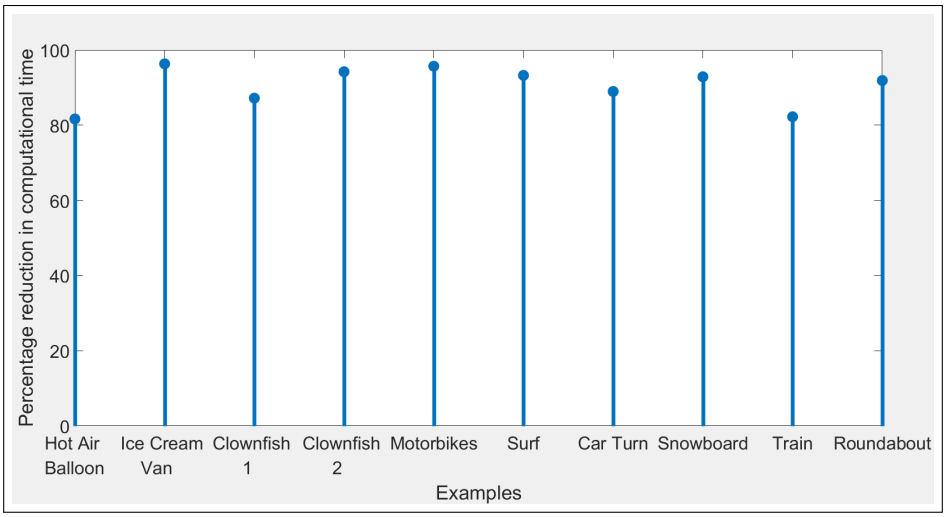


Figure 5.1: Savings in computational time using a three level multiresolution approach expressed as a percentage of the time taken using a purely high resolution approach for different examples in Tables. 3.4 to 3.4

the algorithm is run to convergence at the highest level of resolution and corresponding results are compared using an approach that works at three levels of resolution.

In this multiresolution scheme, the highest level of resolution is level 3. At level 2 the resolution is half of that at level 3 and at level 1 (which is the lowest level of resolution) the resolution is one-fourth of that at level 3. Starting at level 1 the algorithm is initialized and run to convergence. The coordinates of the motion centroid and the scale factors coming from the parameterized motion (see Section. 3.2) are then doubled and the algorithm is rerun to convergence at next higher resolution level. The doubling of motion centroid co-ordinates and scale factors coupled with increasing the resolution level by 1 continues until the algorithm has been run to convergence at the highest level of resolution (which is 3). The percentage reduction in the computational time using this approach (when compared to an approach that only operates at the highest level of resolution) for examples in Tables. 3.4 to 3.4 is shown in Figure. 5.1.

5.1.2 Experimental Results - Improving Robustness

Experimental results showing improvements in robustness using a multiresolution version of the proposed technique are presented here. For each data example an initialization that assumes only the layer count apriori is used. Starting at the highest level of resolution and assuming a two-layered model, a checkered board initialization (similar to that shown in Figure. 2.2) is given to the foreground layer and the algorithm is run to convergence. The experiment is then repeated with this same checkered board initialization but now working through three levels of resolution. Other than the difference in initialization this three-level approach is otherwise identical to the approach discussed in Section. 5.1.1. The final positions of the layer boundaries using both this multiresolution approach and an approach that only works at the highest level of resolution (level 3) are shown for some data examples in Figure. 5.2. It can be observed that the multiresolution scheme enables the algorithm to *wriggle out* of local minimizers that may arise when optimizing purely at the highest level of resolution.

5.2 Parallel Processing of PDEs

This section outlines a method to further improve the computational speed by exploiting the parallel structure of the PDEs involved. As such there are a set of PDEs for each layer that have to be optimized for every iteration of the gradient descent algorithm outlined in Section. 3.1. This can make the issue of computational speed consequential. The most significant PDEs are those for computing the smooth appearance models (Eq. 3.4) and the pointwise Lagrange multiplier functions (Eq. 3.6). These in particular form the bottleneck of runtime speed. Advantage can be taken of the fact that these PDEs are highly parallel in their nature and their numerical solution at various points of the domain over which they are solved can be updated in parallel. Correspondingly, distributing the task of finding their optimizers by using multiple threads to optimize over different subsets of their respective

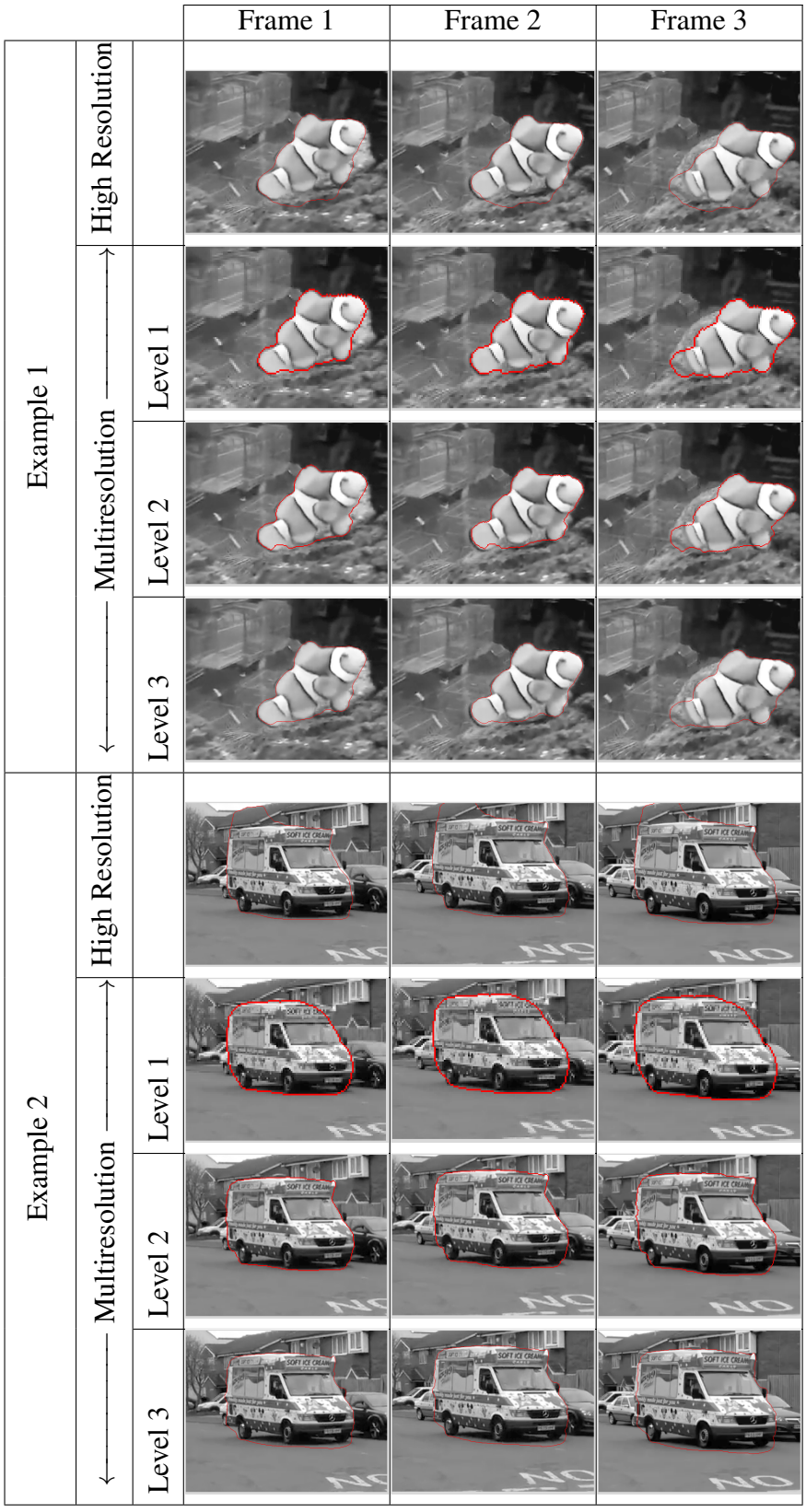


Figure 5.2: Improvements in robustness (resisting local minimizers) using a three-level multiresolution approach for different examples

domains can parallelize the optimization process and lead to faster convergence. This will reduce the time between each iteration of the gradient descent algorithm thereby speeding up the overall performance of the algorithm.

Eq. 5.1 and 5.2 show the update equations for numerically solving the PDEs of Eq. 3.4 and 3.6 respectively. Here Δt and Δx are the time step and space step respectively, used during the discretization process. x and y are the spacial co-ordinates of an arbitrary point on the discretized domain of the respective PDE. The variable t shows a discreet point in time at which the numerical solution for the PDE is updated. By looking at Eq. 5.1 and 5.2 it can be seen that the updated values (for the quantity being optimized) at a particular instant of time only depends on the current values (of that quantity). Therefore the domain of support of the corresponding PDE can be divided into subdomains and at a particular instant of time the updates (over each subdomain) can be computed in parallel. A multithreaded environment is set up to divide the domain of support into subdomains for this purpose (Figure. 5.3).

$$
\begin{aligned}
&f_l^*(x,y,t+1) = f_l^*(x,y,t) \\
&+ \Delta t\big(1-\alpha\big)\sum_{n=1}^{N}\gamma_n V_{l,n}\big(I_n \circ g_{l,n}(x,y,t) - f_l^*(x,y,t)\big) \\
&+ \alpha\frac{\Delta t}{\Delta x^2}\big(f_l^*(x+1,y,t) + f_l^*(x-1,y,t) - 2f_l^*(x,y,t)\big) \\
&+ \alpha\frac{\Delta t}{\Delta x^2}\big(f_l^*(x,y+1,t) + f_l^*(x,y-1,t) - 2f_l^*(x,y,t)\big)
\end{aligned}
\tag{5.1}
$$

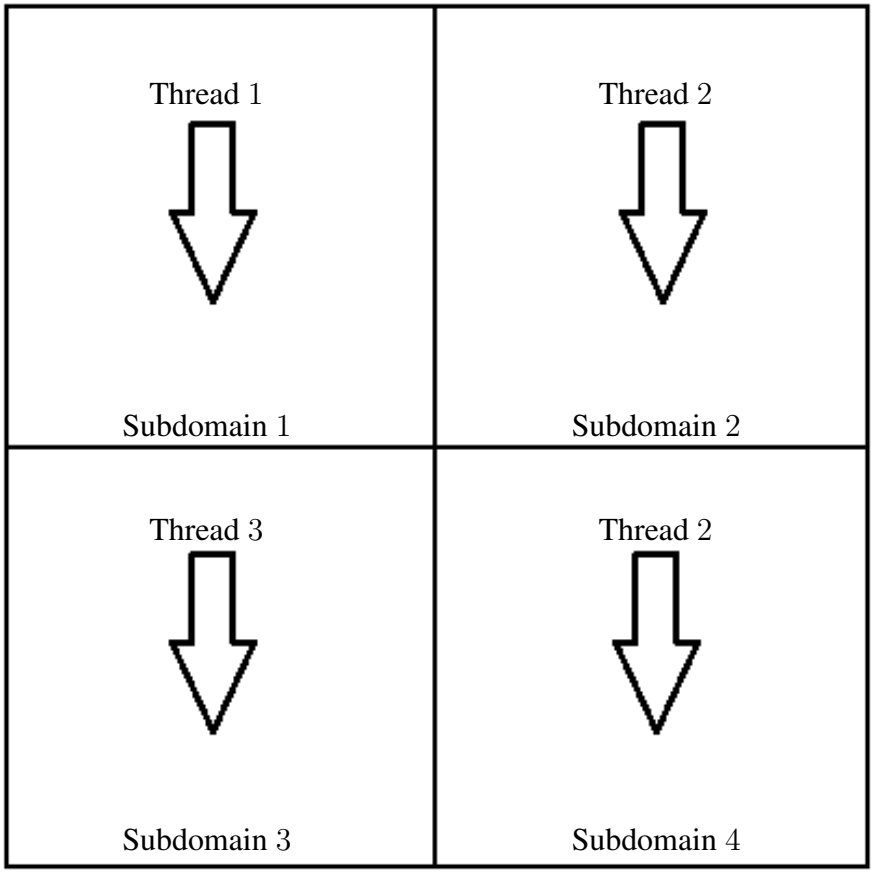


Figure 5.3: Dividing the domain of support of a PDE into subdomains for parallel updating of the solution using a multithreaded environment

$$
\begin{aligned}
&\lambda_l(x,y,t+1) = \lambda_l(x,y,t) \\
&- \Delta t\big(1-\alpha\big)\sum_{n=1}^{N}\gamma_n V_{l,n}\big(\lambda_l(x,y,t) + 2\big(I_n \circ g_{l,n}(x,y,t) - f_l^*(x,y,t)\big)\big) \\
&+ \alpha\frac{\Delta t}{\Delta x^2}\big(\lambda_l(x+1,y,t) + \lambda_l(x-1,y,t) - 2\lambda_l(x,y,t)\big) \\
&+ \alpha\frac{\Delta t}{\Delta x^2}\big(\lambda_l(x,y+1,t) + \lambda_l(x,y-1,t) - 2\lambda_l(x,y,t)\big)
\end{aligned}
\tag{5.2}
$$

Figure. 5.4 shows a plot of the savings in computational time for examples in Table. 3.4 and Table. 3.4 when a multithreaded version of the algorithm was used to parallelize

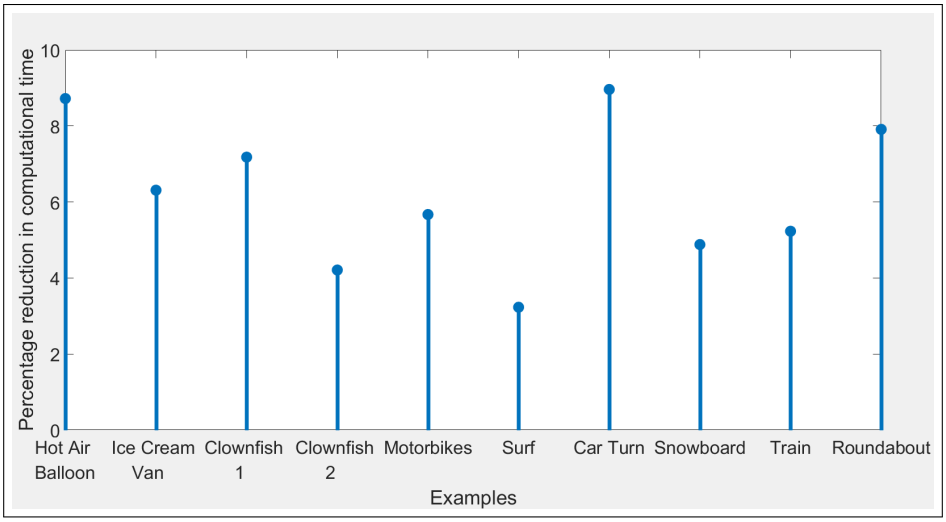


Figure 5.4: Savings in computational time using a multithreaded approach expressed as a percentage of the time taken otherwise for different examples in Tables. 3.4 to 3.4

the optimization process for the PDEs used to compute the smooth appearance models (Eq. 3.4) and the pointwise Lagrange multiplier functions (Eq. 3.6).

It can be observed that the saving in computational time is not immense. This has primarily to do with the fact that multithreading is a much more complex process than what it appears to be at face value. Typically there are overheads for creating and merging threads which can (in some cases significantly) detract from the total savings in computational cost otherwise achieved.

5.3 Strategic Initialization

A strategic initialization (Figure. 5.5) can be helpful in both, reducing the computational time as well as improving the robustness of the algorithm towards local minimizers. The idea is to initialize the algorithm as near as possible to the global minimum. It is clear that this automatically reduces the chances of the algorithm getting stuck in local minima. Being initialized closer to the global minimum it also reduces the time taken to reach it. To strategically initialize the proposed method each active contour involved is initialized as near as possible to the respective target object that it intends to capture. This is done in two

different ways.

Firstly the strategy to discover motion regions (or clusters) using superpixels and textural blocks (Chapter. 4) that was adopted for this very purpose is leveraged. The output from this approach (at a level of resolution which is one-fourth that of the original) is taken. A separate active contour is used for each motion cluster discovered, and bounds the corresponding cluster. The algorithm is run to convergence. The experiment is then again run to convergence but this time using a checkered board initialization for the layers (similar to that shown in Figure. 2.2). The savings in computational time when compared to the checkered board initialization are recorded.

Secondly, each contour is initialized as a rectangle around the target object. The dimensions of the rectangle are chosen to be 20% more than the dimensions of the smallest rectangle that bounds the object. Using this (strategic) initialization the experiment is similarly repeated and the savings in computational time in comparison to the checkered board initialization are recorded. Figure. 5.6 shows a plot of the savings in computational time using a motion cluster based initialization in comparison to a checkered board initialization for examples in Table. 3.4 and Table. 3.4. Figure. 5.7 correspondingly shows a plot of the savings in computational time using a bounding rectangles initialization in comparison to the checkered board initialization.

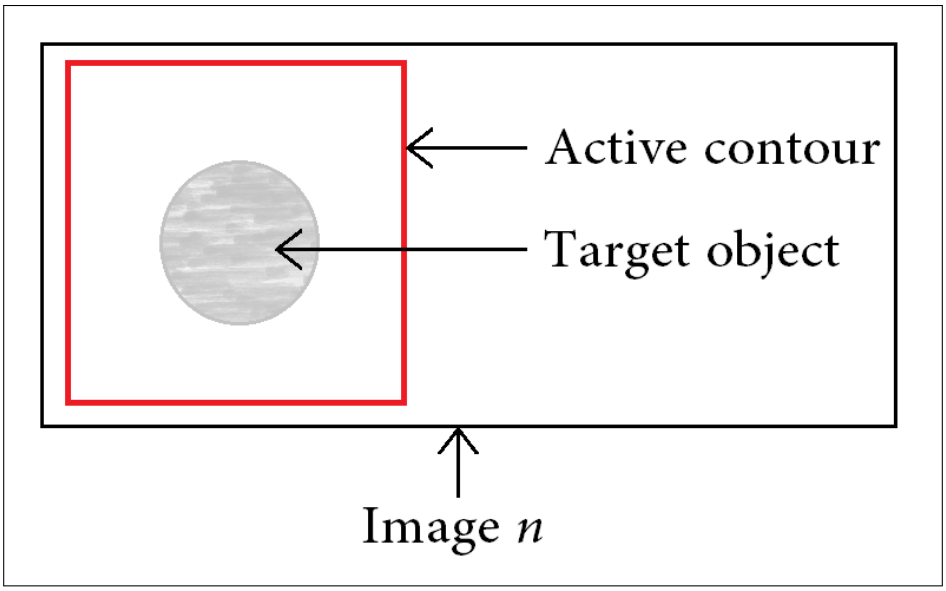


Figure 5.5: Strategically initializing an active contour near the target object

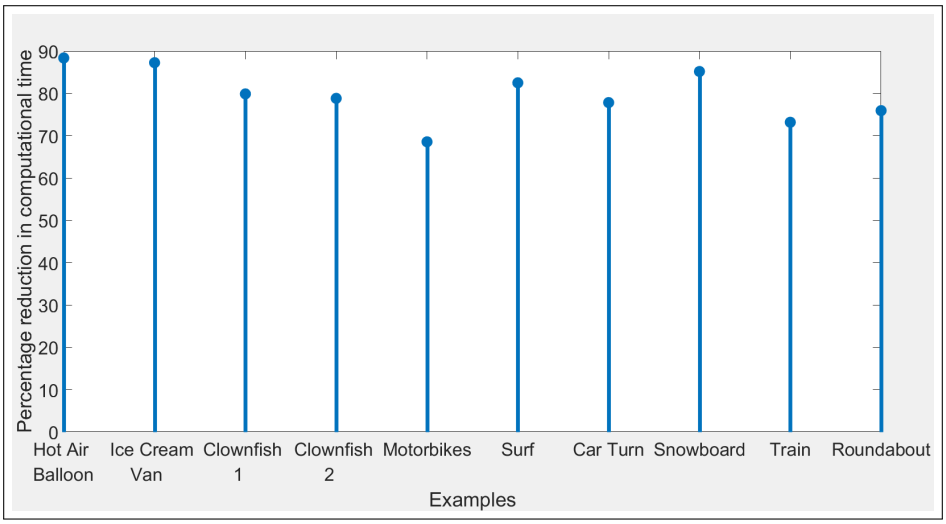


Figure 5.6: Savings in computational time using a strategic initialization for the active contours based on motion clusters discovered using superpixels and textural blocks (Chapter. 4) expressed as a percentage of the time taken using a checkered board initialization for different examples in Tables. 3.4 to 3.4

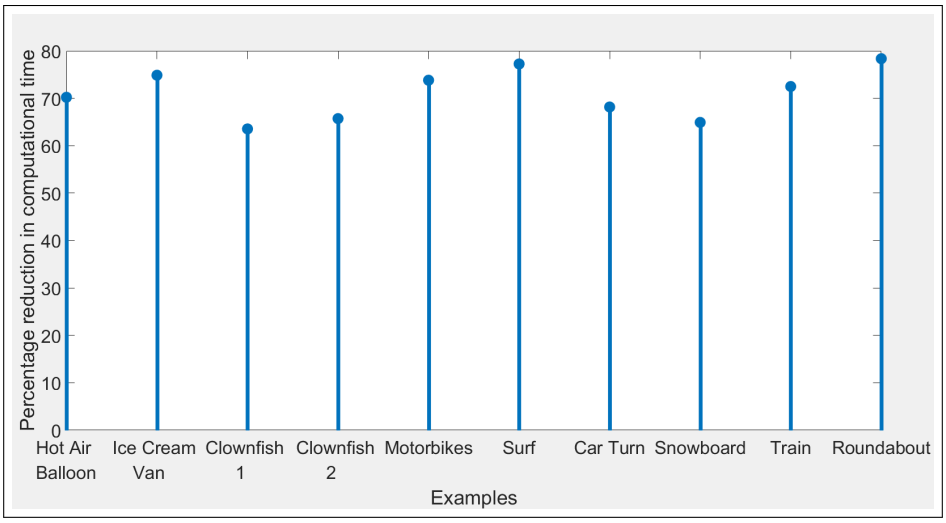


Figure 5.7: Savings in computational time using a strategic initialization for the active contours based on bounding rectangles around target objects expressed as a percentage of the time taken using a checkered board initialization for different examples in Tables. 3.4 to 3.4

CHAPTER 6

APPLICATIONS:- VIDEO COMPRESSION

One particular application where potential for the proposed model is anticipated is video compression. Videos typically consist of sequences of images (or frames) that exhibit a huge amount of (temporal) redundancy. More specifically the redundancy in these (successive) frames typically manifests as a finite number of objects that relocate themselves within the frames. The objects usually have a certain ordering for the occlusions between them. This structure can be aptly captured by the generative layered model proposed in Section. 3.1. While the entire video sequence may not be able to be represented this way there are likely to be a number of subvolumes within the sequence that can be reasonably approximated in this manner.

Different layers can be used to capture the shape and appearance of the various objects within the subvolume (with one layer being used to capture the background). The depth ordering of the layers can take into account the occlusion structure and the motion models for the layers can be used to hold the relocation information of the objects. Once the layered models have been computed the additional overhead for resynthesizing each frame in the subvolume is just L vectors in $\mathbb{R}_6$ where L is the total number of layers used to represent the subvolume. This could lead to an efficient representation of a number of subvolumes in many video sequences. The proposed method is used to store and resynthesize certain subvolumes of video sequences and the results are compared to the technique used in the relevant subsystem in the state of the art technology [49, 50]. Visually observable results and comparisons of benchmarks such as PSNR, MAD and SSIM [51, 52, 53] used to measure image and video quality are presented.

6.1 Representing Video in Terms of Moving Objects

The variational framework provided has flexibility in modeling shape, motion, appearance and occlusion structure. It has the necessary infrastructure to synthesize several images in a set, in particular when there are a limited number of moving objects in the images that follow a specific occlusion structure. The primary target is scenarios consisting of independently moving objects exhibiting nearly affine motion. Images representing such scenarios can be decomposed in terms of the various moving objects (that are common to them) coupled with a set of (simplistic) motion models. The images can then be resynthesized using the objects and motion models. This would allow for a highly efficient as well as a reasonably accurate representation of the images. Typically such sets of images can be found in video sequences.

A highly efficient storage technique for such a group of images based on this philosophy is proposed, that uses the proposed variational framework outlined earlier (Section. 3.1). The technique called MOVE (Moving Object Video Encoding) is introduced and encodes a set of images in terms of its moving objects.

6.2 Macro Blocks versus Layers

Most video sequences typically have subsets of 15 to 50 frames in which the scene does not change a lot. The motion exhibited between the frames is usually minimal and can be aptly captured with a set of affine motion models (that allow for rotation, scaling and translation of the moving objects or background). The state of the art techniques (such as HEVC or H.265) [49, 50] use a translation (motion) model to locate finite sized blocks (called macroblocks) of one frame in another. Here a particular frame in the subset (also called the I frame) is chosen as the reference frame. The blocks in the other frames (also called P frames) are then relocated in the I frame using this translational model (Figure. 6.1). The translations obtained are typically called motion vectors. The P frames can then

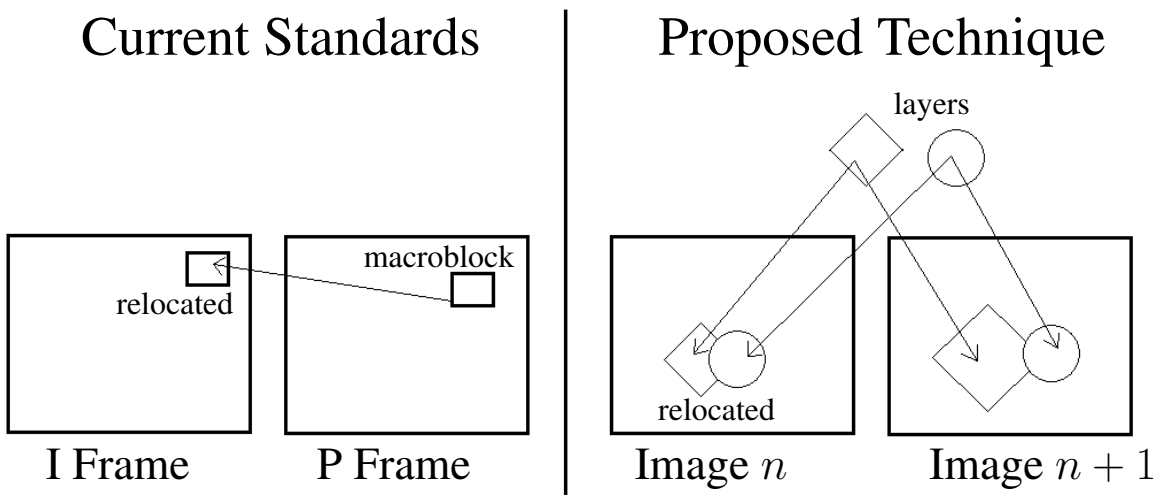


Figure 6.1: Image resynthesis using macroblocks versus the proposed technique

be resynthesized using the I frames in conjunction with the motion vectors. This technique sometimes creates (minor yet noticeable) artifacts and may also give a 'blocky appearance' to the resynthesized frames.

The frames can alternately be resynthesized using the proposed variational framework outlined in Section. 3.1. The appearance models for the layers along with their motion models are used. These motion models now allow for rotation and scaling of the layers in addition to translation. Furthermore, the usage of active contours offers richness in modeling arbitrary shape and topology. As such they can more accurately capture the geometry of moving objects and are likely to reduce the blockiness seen in the resynthesized images using current standards. Correspondingly, there is potential in the generative layered model proposed for providing an alternative and improved method for video compression. Figure. 6.1 shows the differences in the resynthesis strategies.

6.3 Smooth Appearance Models vs Image Mapped Models

One of the salient features of the proposed technique is the relaxation of the brightness constancy constraint when tracking motion in a set of images. This makes the model more

robust to tracking moving objects in most real life scenarios. A downside of this is that the layers (which represent the moving objects) experience a smoothing effect or 'blur' in their appearances which becomes visible when the layers are used to resynthesize the images. To overcome this the smooth appearance models of the layers are replaced with 'image mapped' models (explained ahead) once the layer motion has been determined. The image mapped layers are then used to resynthesize the images.

The problem of blurriness can be explained by assessing how the relaxation of the brightness constancy constraint manifests itself in the layer appearances. There are two ways, *interframe smoothing* and *intraframe smoothing* by which this happens (Figure. 6.2).

Firstly, the layer appearance at a particular point 'x' is influenced by data from all those images in which that point (when mapped) is visible. This produces an effect of averaging the appearance at the layer point 'x' based on multiple data values coming from (the mapped points $g_{l,n}(x), n \in [1\ N]$ in) the different images (interframe smoothing). The result is a loss in accuracy when this point (x) is later on used to approximate the corresponding (mapped) points in each of the images.

Secondly, the layer appearance at a particular point is also influenced by the appearance of the neighboring points within that layer (intraframe smoothing). The overall result is a blurriness in the resynthesized frames.

The problem can be significantly mitigated by completely omitting intraframe smoothing and choosing only one image to model a particular point of the layer when that point is visible in multiple images. To select the (one) image for modeling a particular layer point (x) a natural thing to do is to select the image whose intensity at the correspondingly visible point ($g_{l,n}(x)$) is closest to that of the smooth appearance model for the layer in question at that point (x). However it was found that in practice the following strategy works better. First a local search patch of 7x7 pixels is made in each image in which the point (x) is correspondingly visible. The patch in a particular image (image n) is centered around the correspondingly visible (or mapped) point ($g_{l,n}(x)$). The number of points within that patch

belonging to the layer in question are counted and the image with the maximum count is selected. The intensity (or appearance) of the correspondingly visible point in the selected image is then used as the intensity of the layer point (x). This causes the image with the maximum local support for a particular layer to be selected for modeling its appearance. If there is a tie in this count between more than one image then the image whose intensity at the correspondingly visible point is closest to that of the smooth appearance model at the layer point (x) is selected. In this manner the appearance for each point of the layer's domain is replaced with the appearance of a mapped point from one of the images. The technique called image mapped layers is illustrated in Figure. 6.2. The figure shows image $n+1$ being selected to model the appearance of layer l at the point x. The intensity (appearance) of the layer at x is set equal to the intensity of image $n+1$ at $g_{l,n+1}(x)$.

6.4 Benchmarks Used for Image and Video Quality

This section touches on selected benchmarks used to assess the quality of reconstructed images by the proposed technique. Three popular benchmarks, the SSIM (Structural SIMilarity), PSNR (Peak Signal to Noise Ratio) and MAD (Mean Absolute Distortion) are evaluated. Eq. 6.1 to 6.3 give the mathematical expressions for these benchmarks.

$$SSIM(X,Y) = \frac{\left(2\mu_X\mu_Y + c_1\right)\left(2\sigma_{XY} + c_2\right)}{\left(\mu_X^2 + \mu_Y^2 + c_1\right)\left(\sigma_X^2 + \sigma_Y^2 + c_2\right)} \tag{6.1}$$

$$PSNR(X,Y) = 10\log_{10}\left(\frac{\left(RANGE\right)^2}{MSE(X,Y)}\right) \tag{6.2}$$

$$MAD(X,Y) = \frac{1}{N}\sum_{i=1}^{N}|X_i - Y_i| \tag{6.3}$$

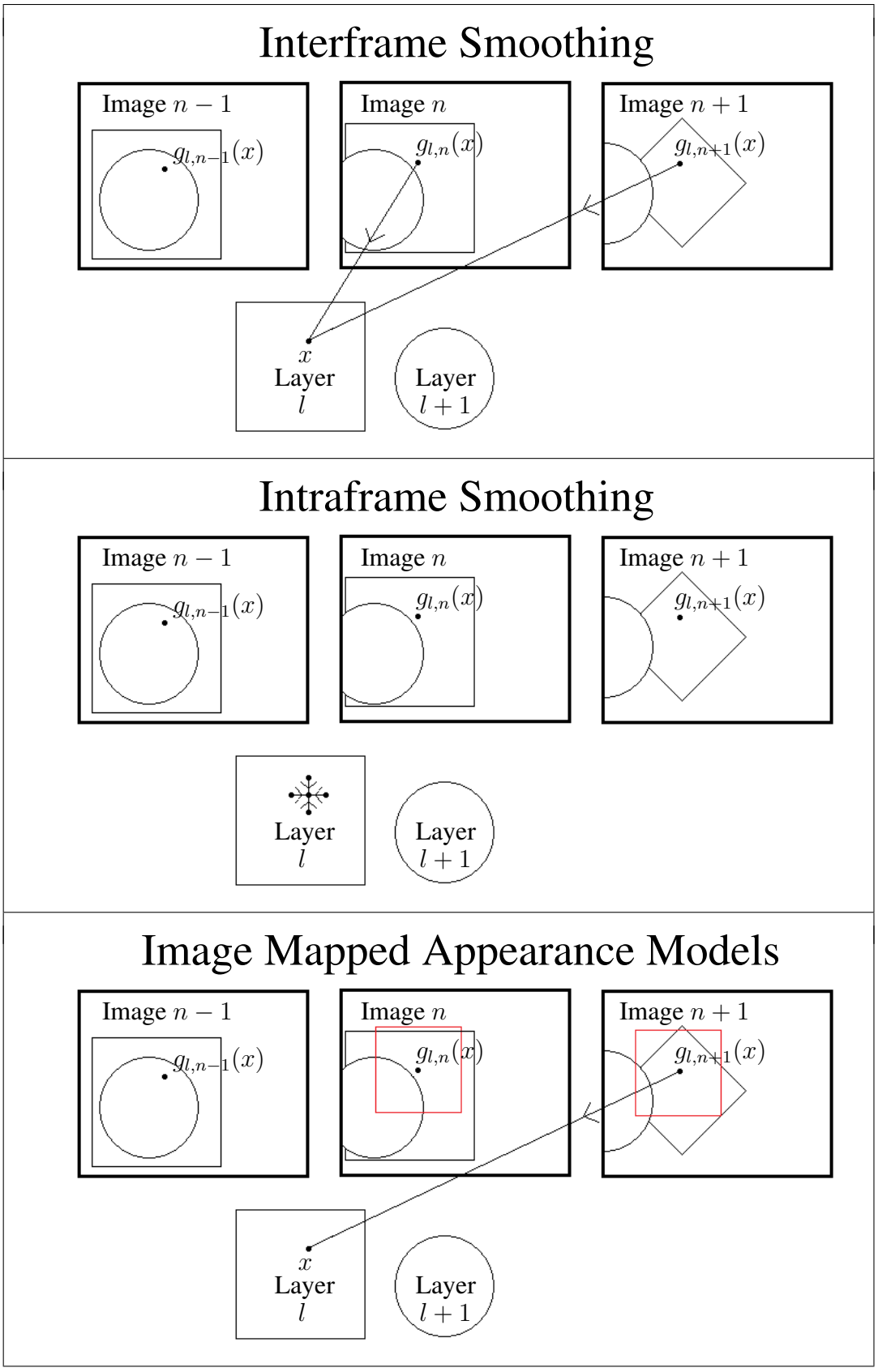


Figure 6.2: Smooth layers vs image mapped layers. Local search patches shown in red.

where X and Y are the original and reconstructed images respectively, N is the total number of (discreet) points in each image, $RANGE = 255$ is the intensity range of the images, $c_1 = 0.0001(RANGE)^2$, $c_2 = 0.0009(RANGE)^2$ and

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} X_i$$

$$\mu_Y = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

$$\sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N} \big(X_i - \mu_X\big)^2$$

$$\sigma_Y^2 = \frac{1}{N}\sum_{i=1}^{N} \big(Y_i - \mu_Y\big)^2$$

$$\sigma_{XY} = \frac{1}{N}\sum_{i=1}^{N} \big(X_i - \mu_X\big)\big(Y_i - \mu_Y\big)$$

$$MSE(X,Y) = \frac{1}{N}\sum_{i=1}^{N} \big(X_i - Y_i\big)^2$$

Higher values for the SSIM and PSNR and lower values for the MAD indicate a better score.

6.5 MOVE (Moving Object Video Encoding) - A Novel Approach for Representing Video

A summary of the novel approach proposed to represent a subvolume within a video sequence is presented. The subvolume is first decomposed into a set of layers and motion models (Figure. 6.3) using the technique outlined in Section. 3.1. The smooth appearances of the layers are then replaced with image values (Section. 6.3) to get image mapped layers. Each layer now (typically) models an independently moving object in the subvolume. The layer encodes its shape, appearance and occlusion information. In addition to this there is

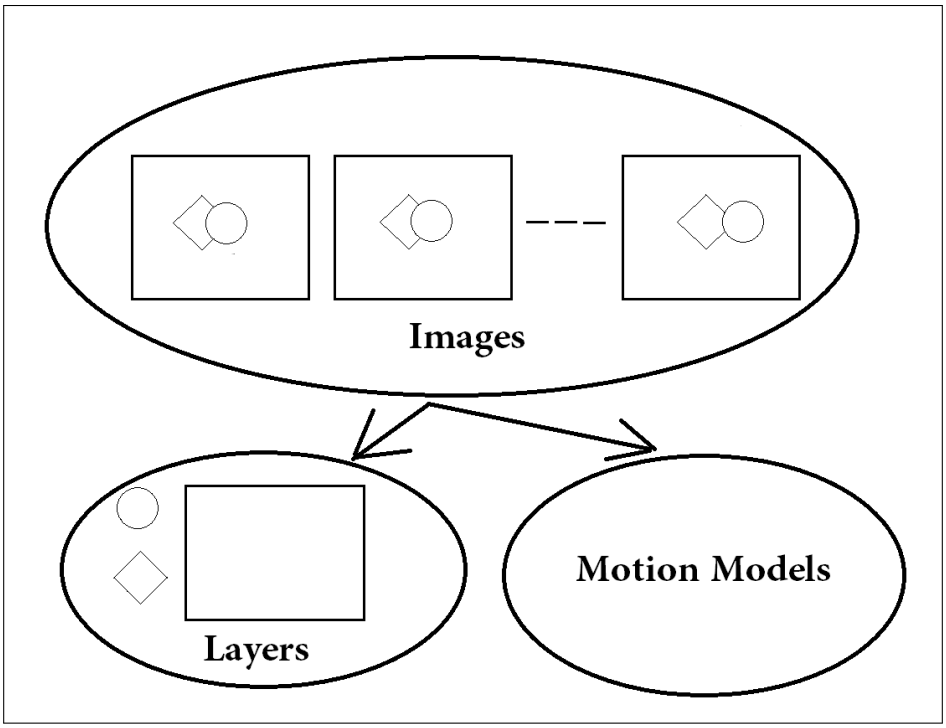


Figure 6.3: Layered representation of a set of images

a set of motion models associated with each layer. These are used in conjunction with the layers to resynthesize the images in the subvolume. Eq. 6.4 gives the equation for resynthesizing the image I_n at x (which is an arbitrary point in I_n) in terms of the layer appearance functions ($f_l, l \in [1\ L]$) and the set of motion models ($g_{l,n}, l \in [1\ L], n \in [1\ N]$). All the points in each image can now be resynthesized using Eq. 6.4. The technique essentially resynthesizes the images using a set of independently moving objects and a set of motion models for those objects. The name 'MOVE' (*Moving Object Video Encoding*) is coined for the technique.

$$I_n(x) = \sum_{l=1}^{L} V_{l,n}\left(f_l \circ g_{l,n}^{-1}(x)\right) \tag{6.4}$$

6.6 Experimental Results: Video Compression

In this section experimental results are presented showing resynthesized or reconstructed images using MOVE, the proposed technique. Results are both qualitatively and quantitatively compared with the macroblock based reconstruction that is used in current video compression standards such as H.265 [50]. Standard benchmarks for estimating the perceived quality of images or videos are evaluated and compared for different datasets (Figure. 6.4 to 6.6). The benchmarks evaluated are the SSIM, PSNR and MAD (see Section. 6.4). In addition to this these benchmarks are evaluated for the first 100 frames of these datasets (Tables. 6.1 to 6.3). It may be noteworthy that image quality assessment is a complicated issue in itself and typical benchmarks may not always be consistent with the ratings given by a human subject. The SSIM (Structural SIMilarity) was designed to closely mimic the score given by a human observer and is widely accepted amongst the video compression community. The other two benchmarks can also be insightful in assessing image and video quality.

For a visual assessment the reconstructed images using the different techniques are also presented (Table. 6.4). The reconstructed images using a macroblock based reconstruction (H.265), the smooth layer models (Section. 6.3) and MOVE (the proposed technique) are shown.

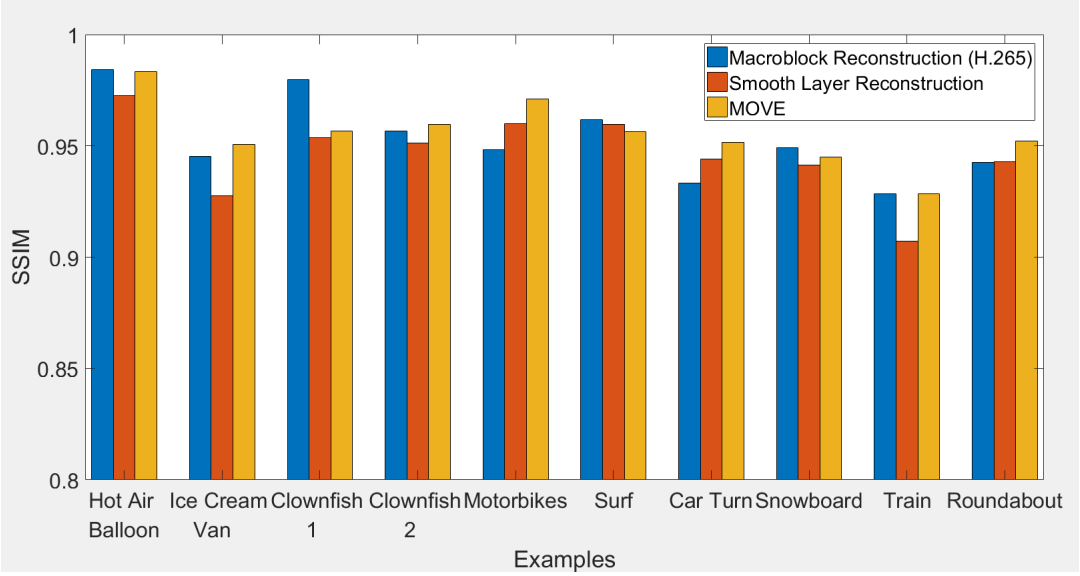


Figure 6.4: A comparison of the average SSIM index for reconstructed frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction, a smooth layer reconstruction and the proposed technique MOVE

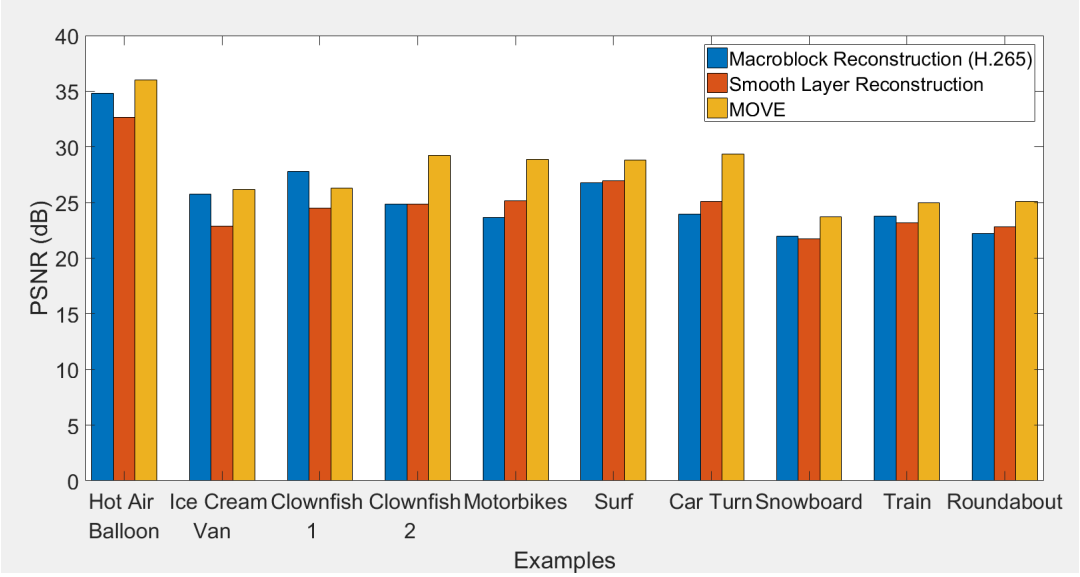


Figure 6.5: A comparison of the average PSNR for reconstructed frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction, a smooth layer reconstruction and the proposed technique MOVE

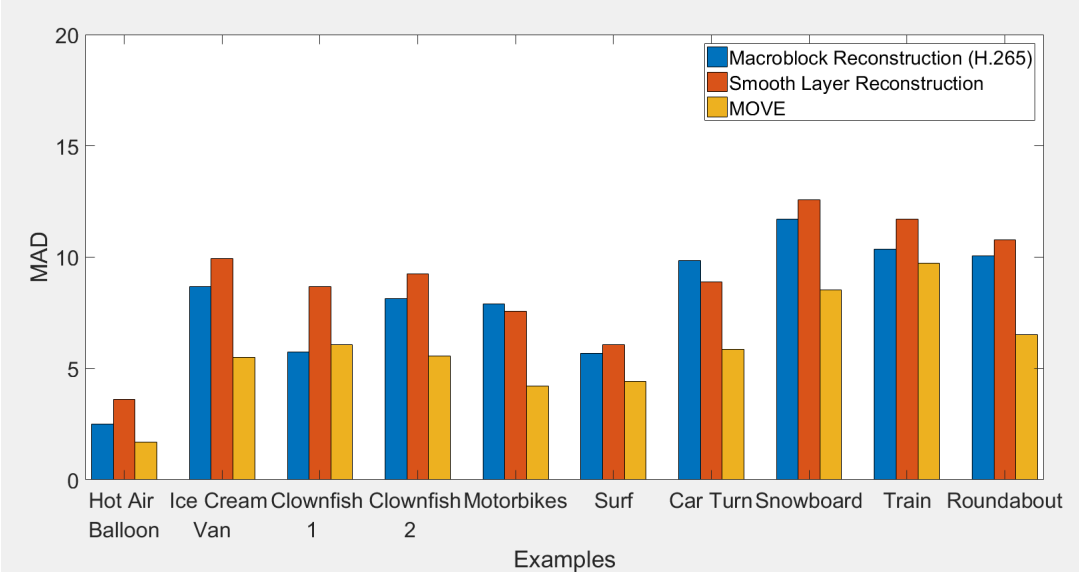
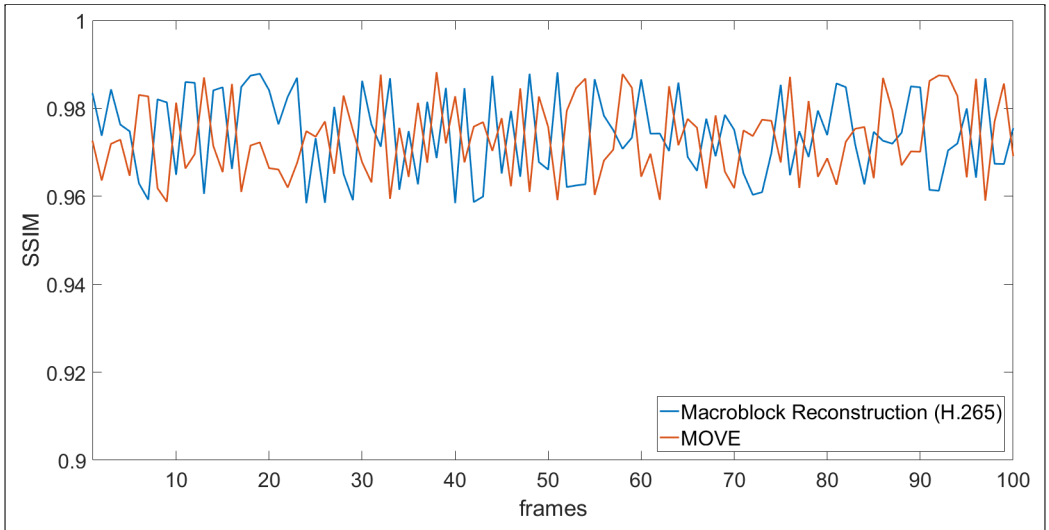
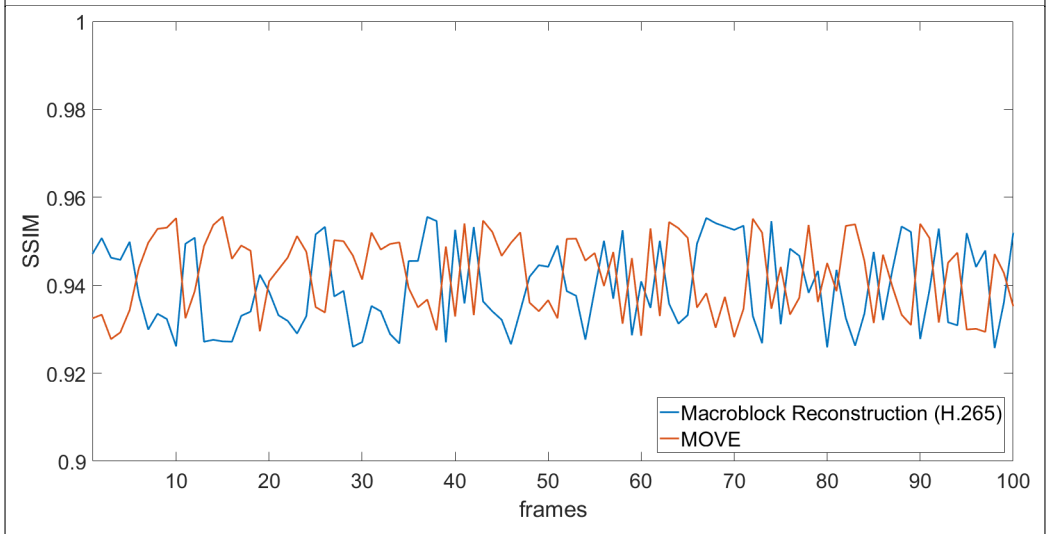


Figure 6.6: A comparison of the average MAD for reconstructed frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction, a smooth layer reconstruction and the proposed technique MOVE

Table 6.1: A comparison of the framewise SSIM for the first 100 resynthesized frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction like that in used in H.265 and the proposed technique MOVE

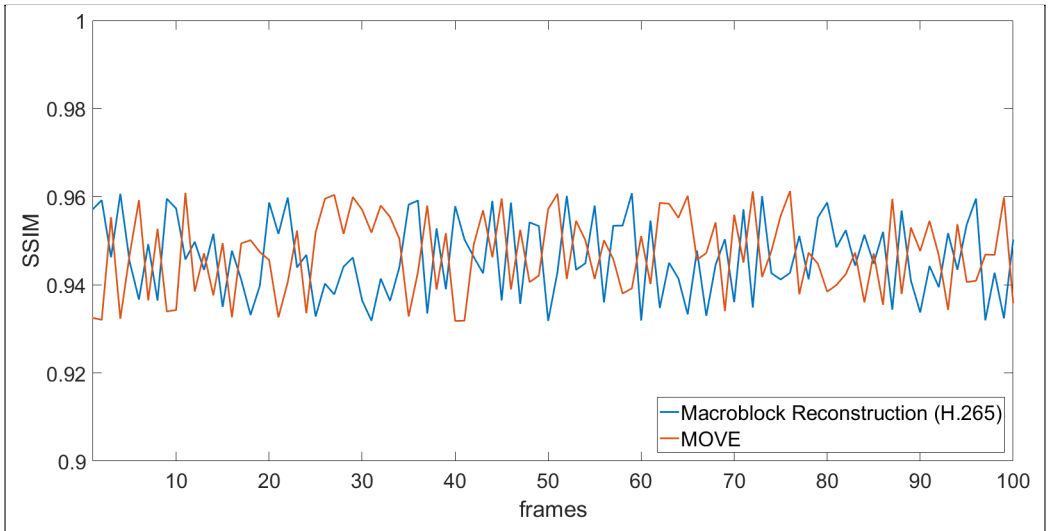


Hot Air Balloon

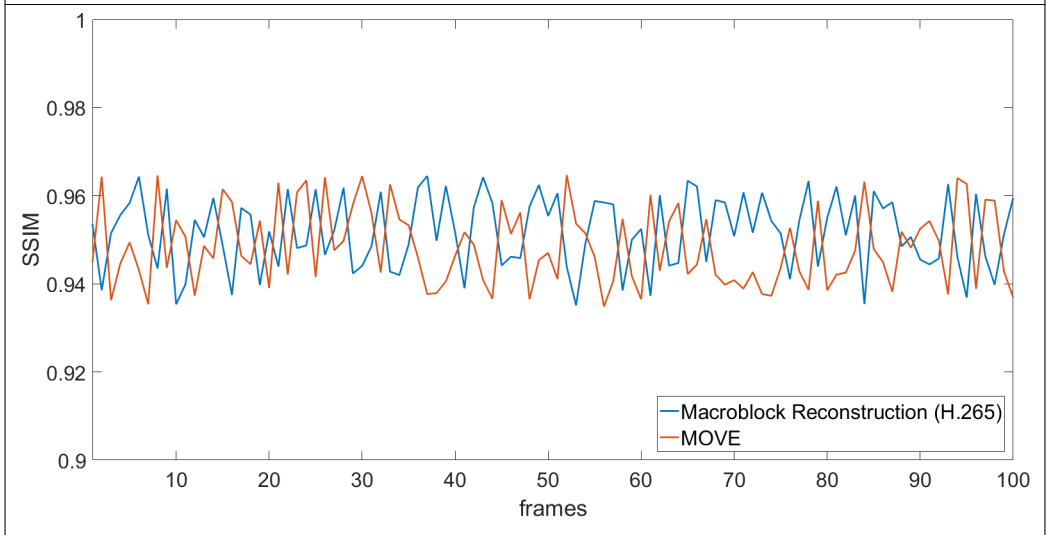


Ice Cream Van

Table. 6.1 Continued on next page . . .

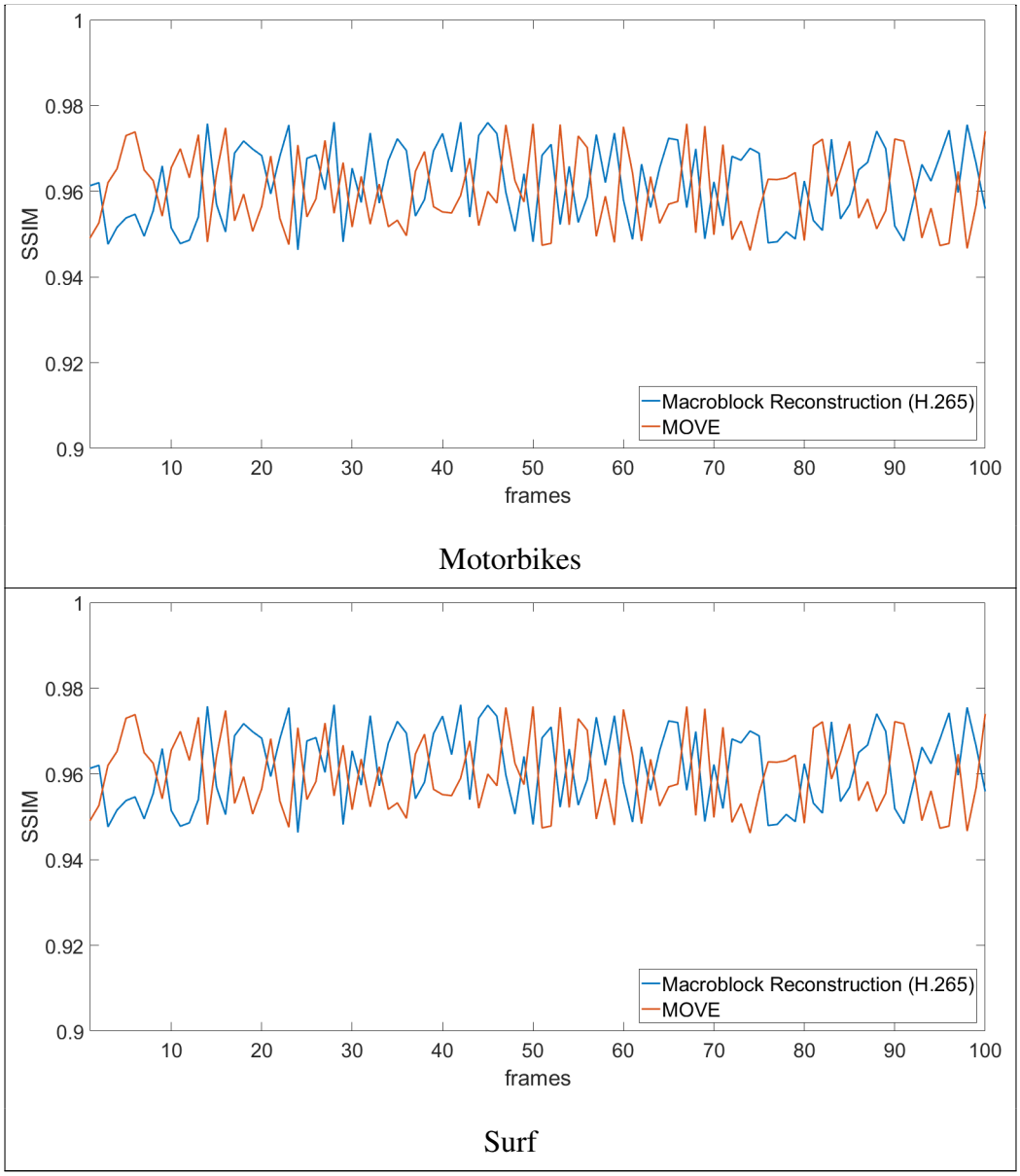


Clownfish1



Clownfish2

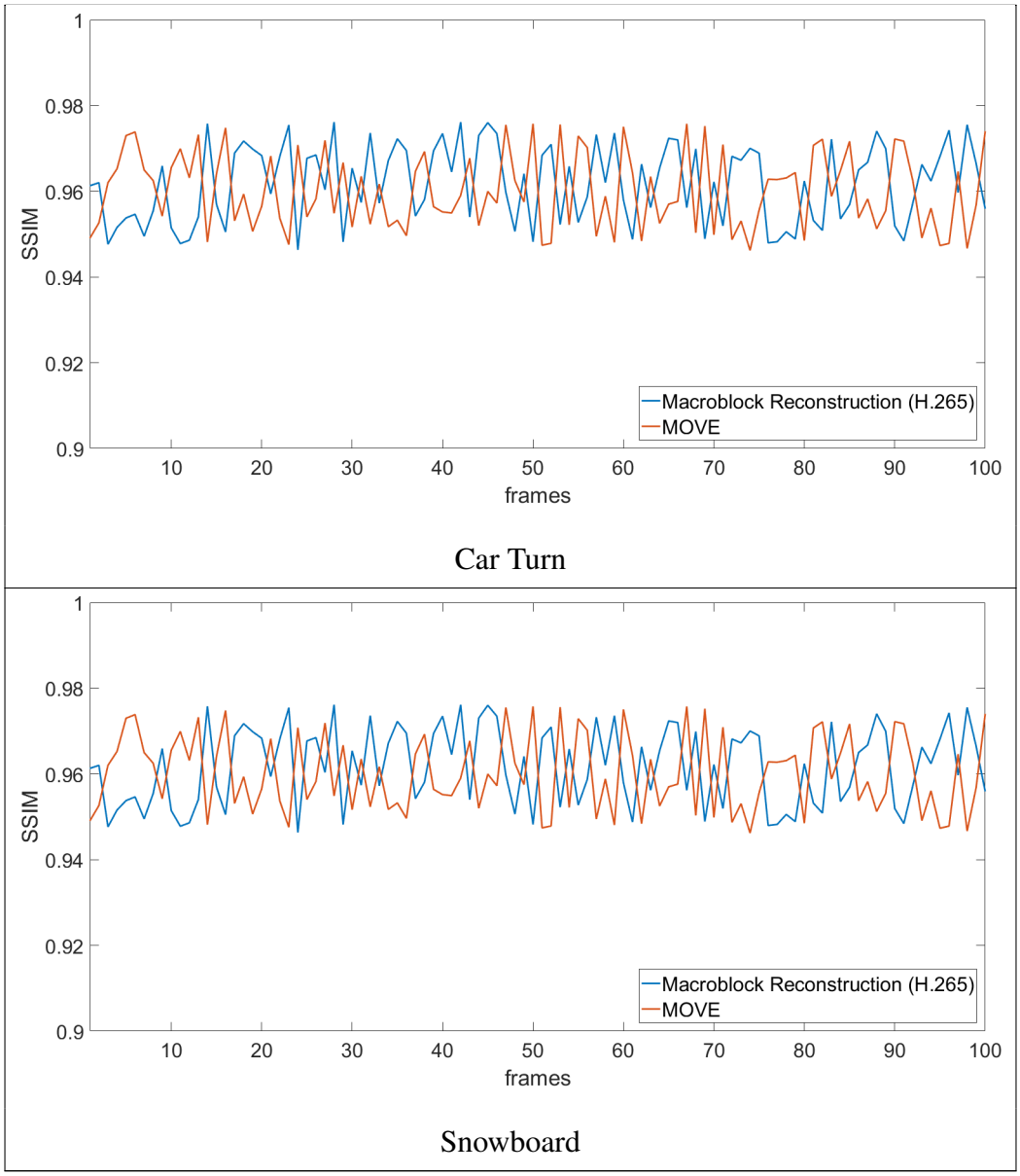
Table. 6.1 Continued on next page . . .



Motorbikes

Surf

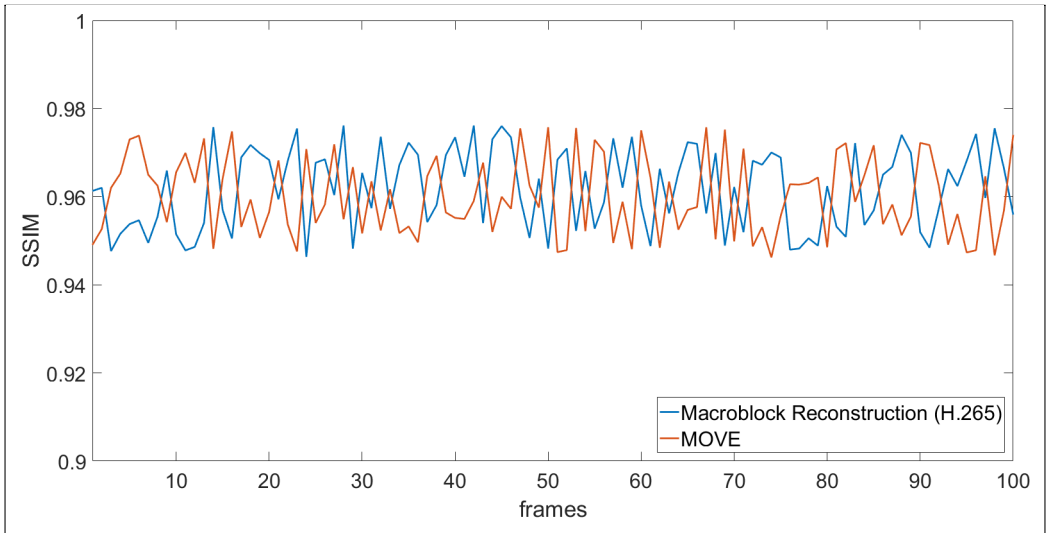
Table. 6.1 Continued on next page . . .



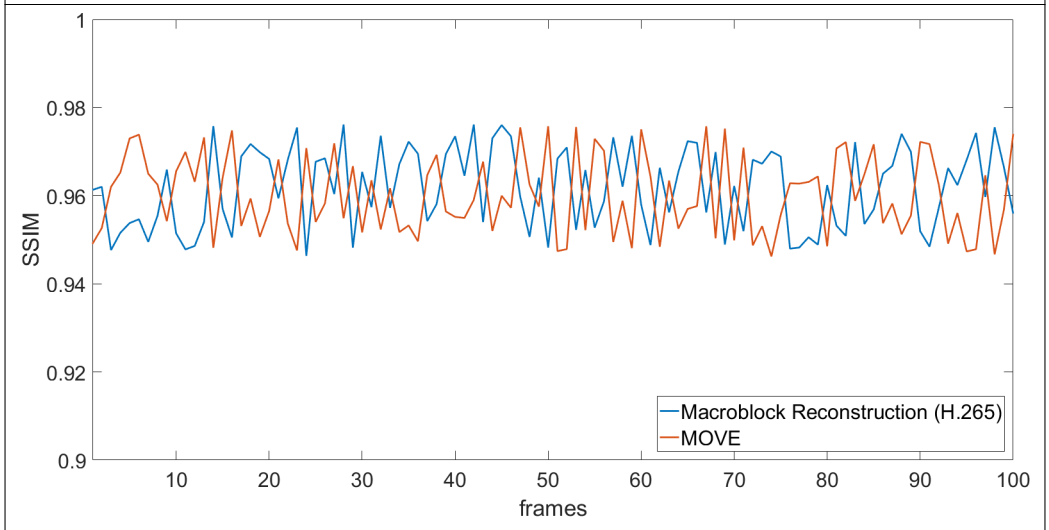
Car Turn

Snowboard

Table. 6.1 Continued on next page . . .



Train



Roundabout

Table 6.2: A comparison of the framewise PSNR for the first 100 resynthesized frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction like that in used in H.265 and the proposed technique MOVE

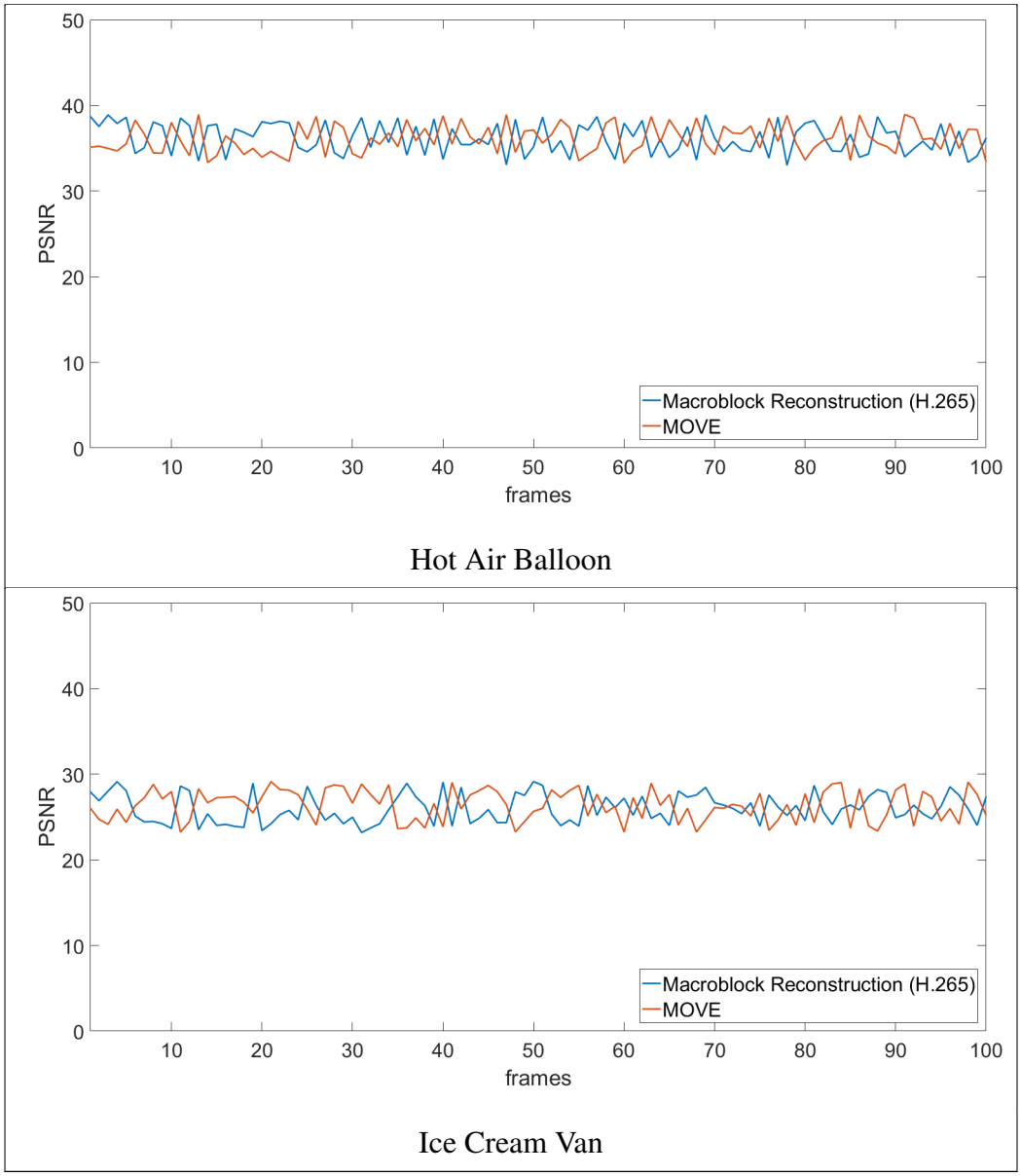
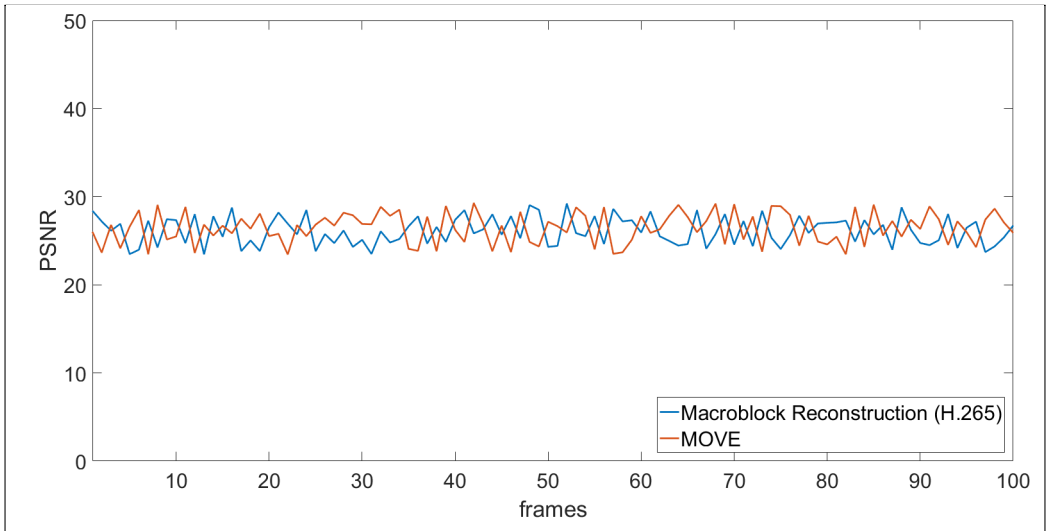
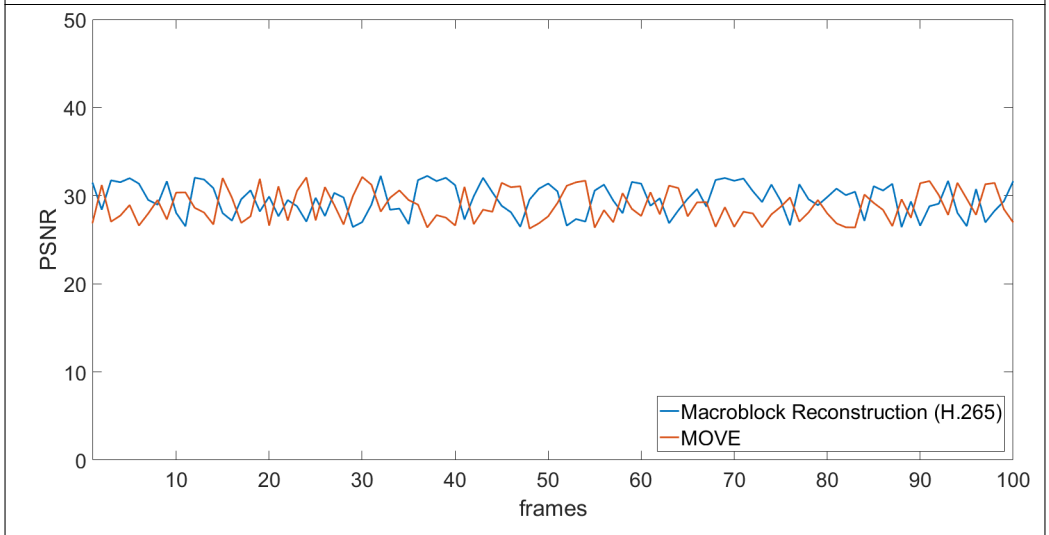


Table. 6.2 Continued on next page . . .



Clownfish1



Clownfish2

Table. 6.2 Continued on next page . . .

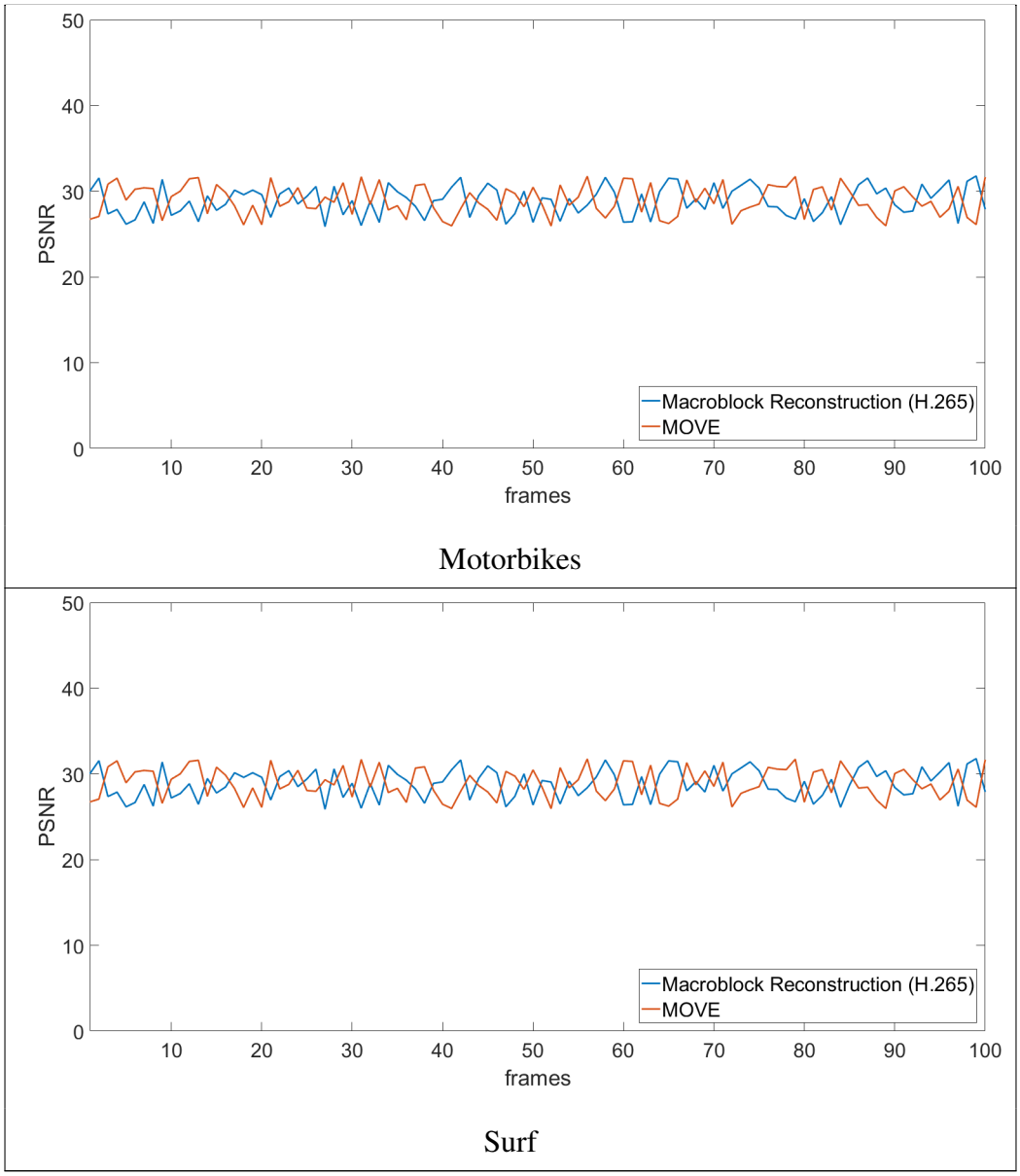


Table. 6.2 Continued on next page . . .

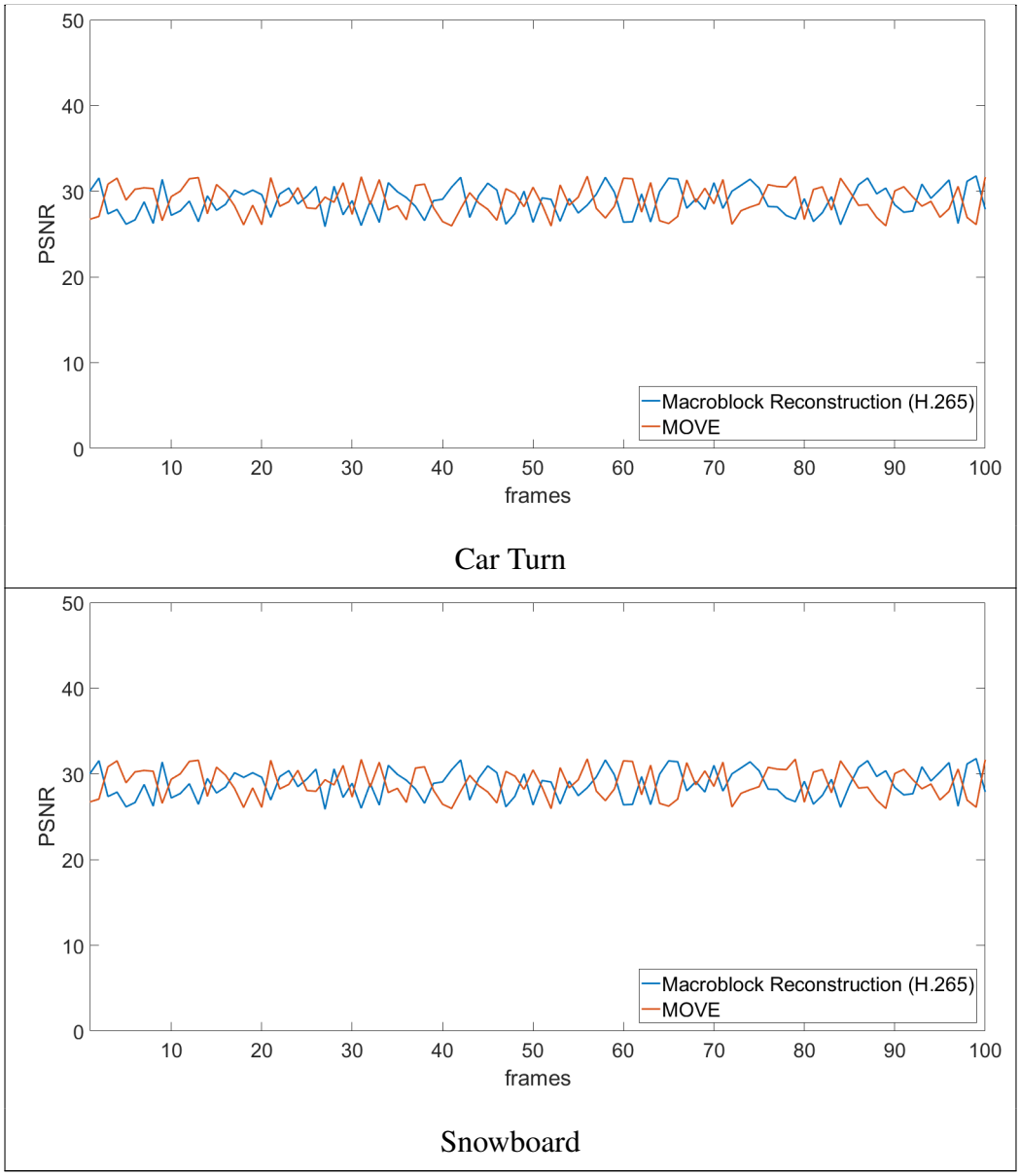


Table. 6.2 Continued on next page . . .

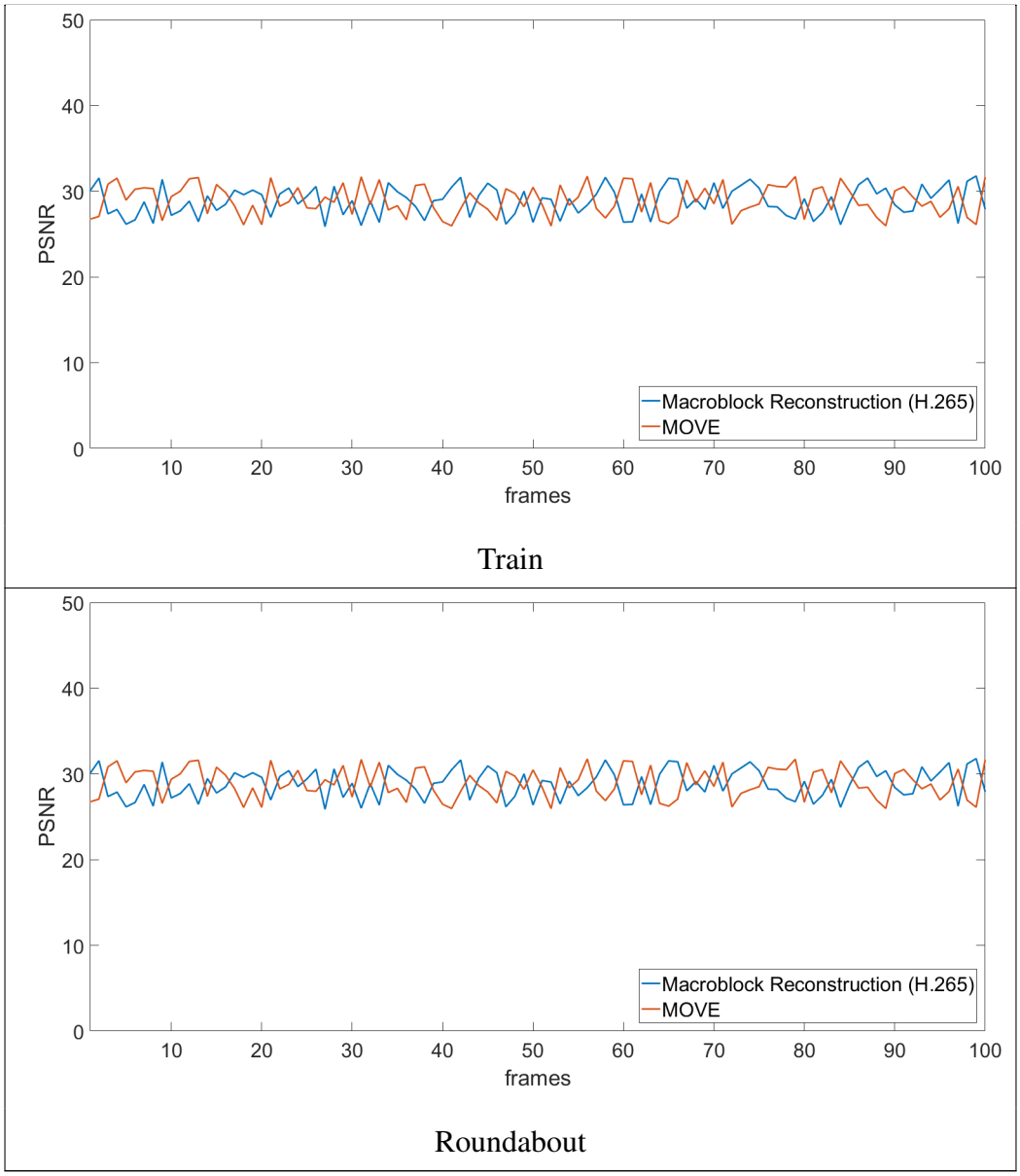

PSNR
frames
Macroblock Reconstruction (H.265)
MOVE
Train
PSNR
frames
Macroblock Reconstruction (H.265)
MOVE
Roundabout

Table 6.3: A comparison of the framewise MAD for the first 100 resynthesized frames of examples taken from Tables. 3.4 to 3.4 using a macroblock reconstruction like that in used in H.265 and the proposed technique MOVE

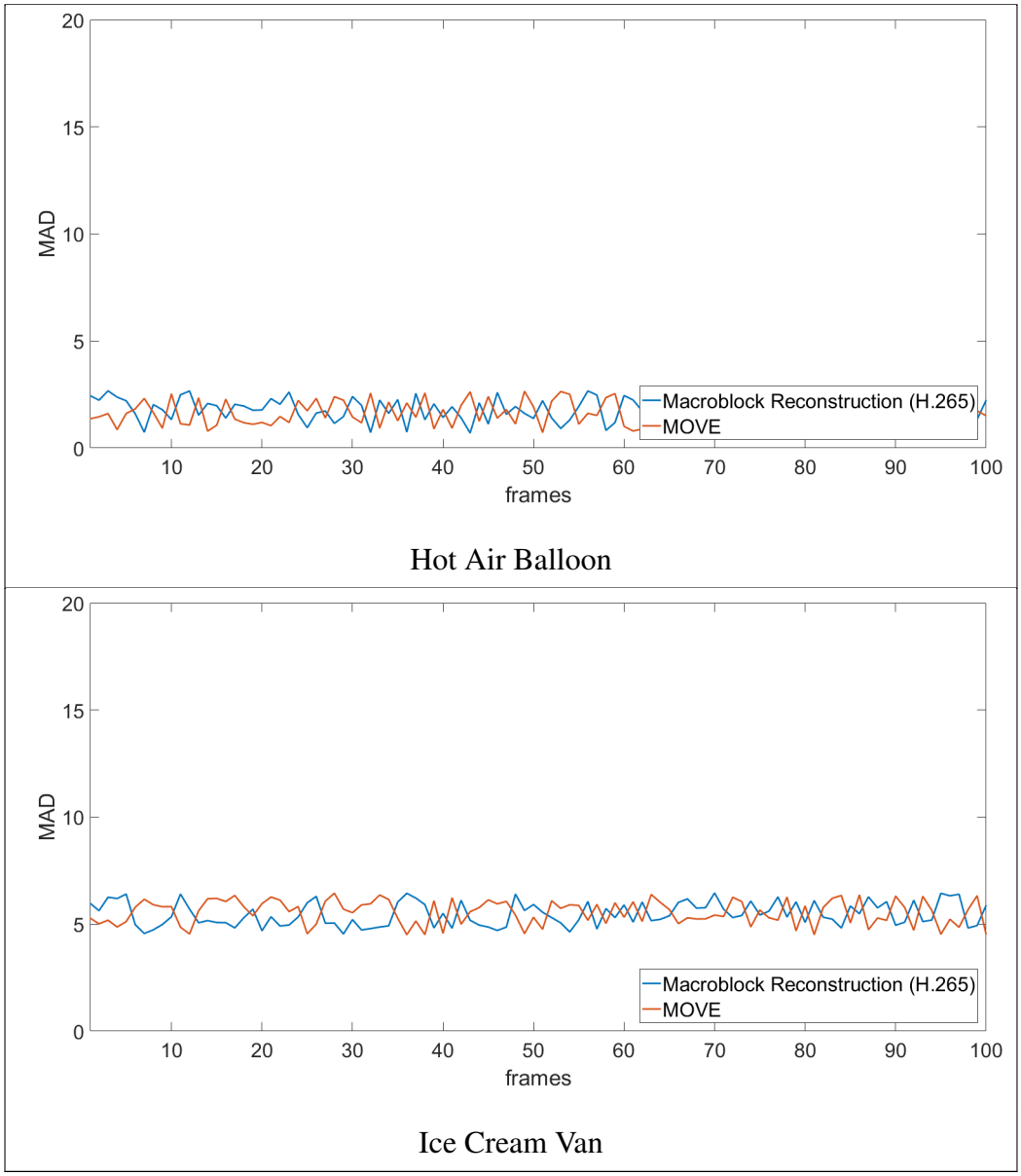
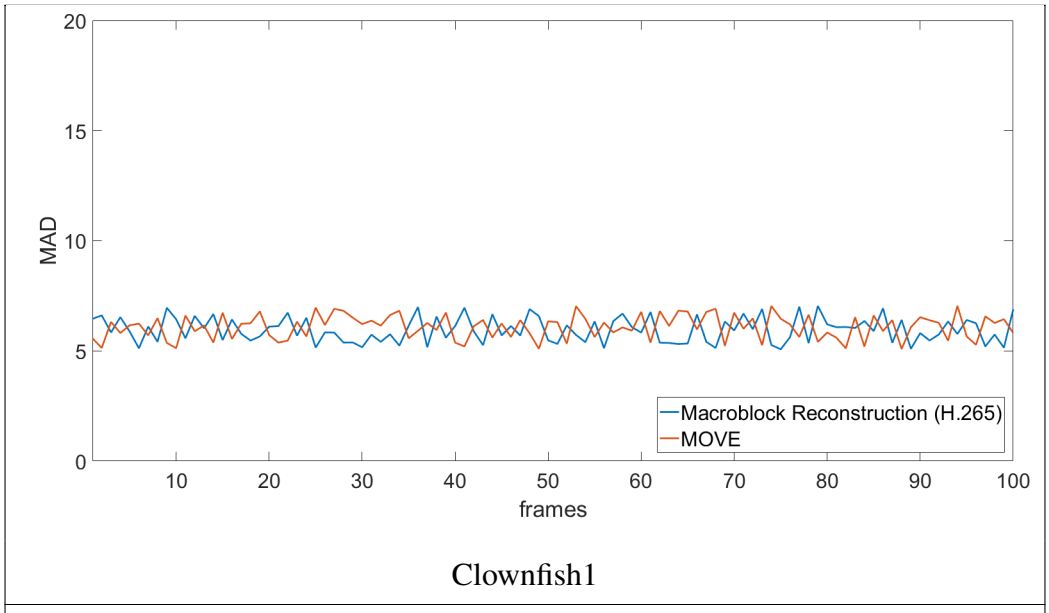
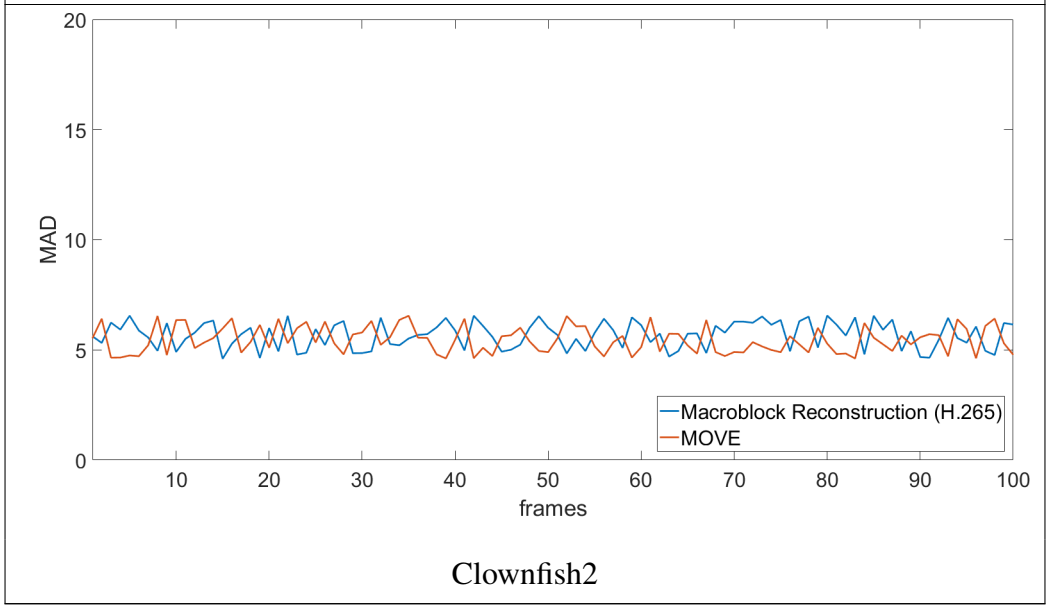


Table. 6.3 Continued on next page . . .



Clownfish1



Clownfish2

Table. 6.3 Continued on next page . . .

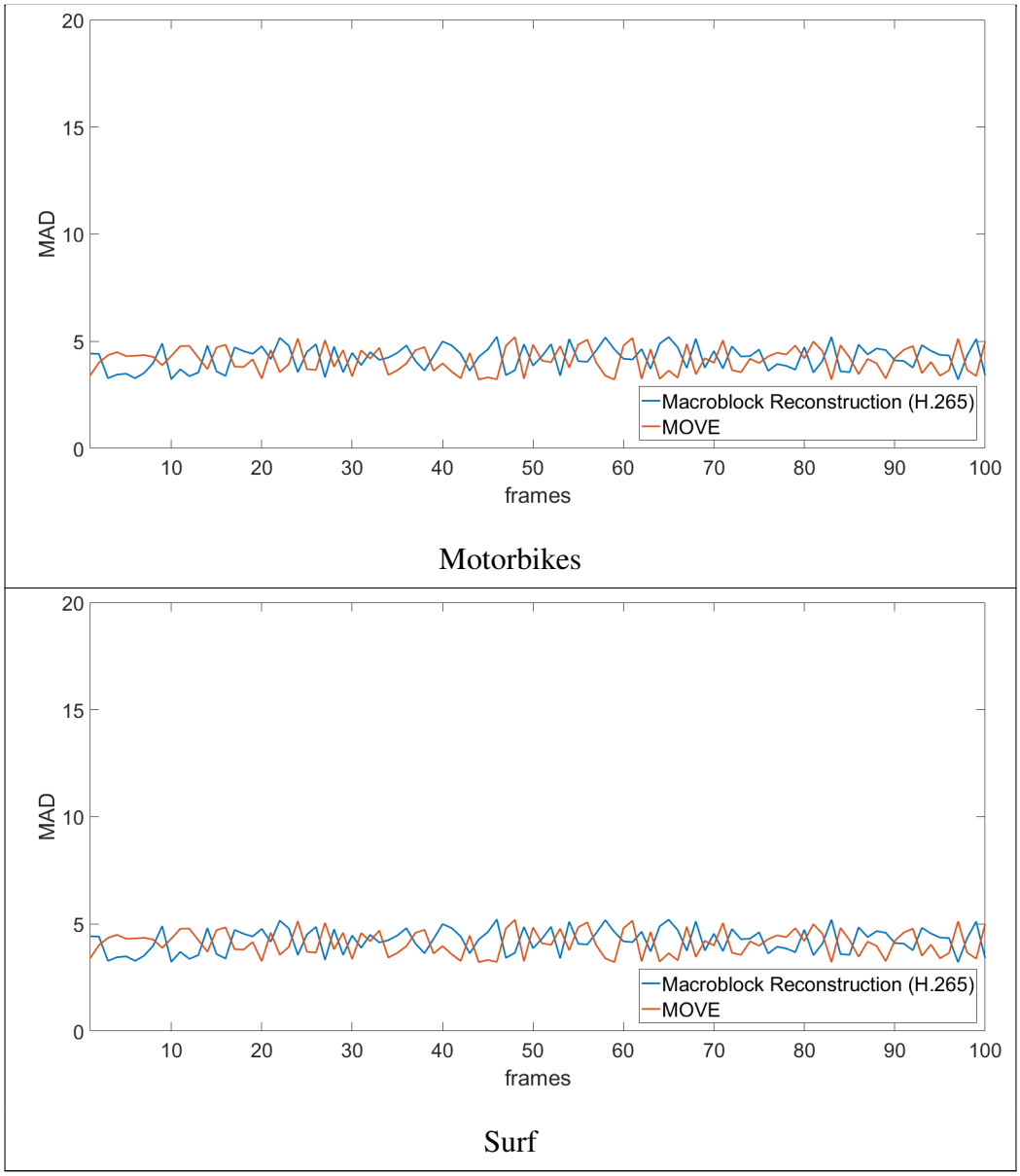


Table. 6.3 Continued on next page . . .

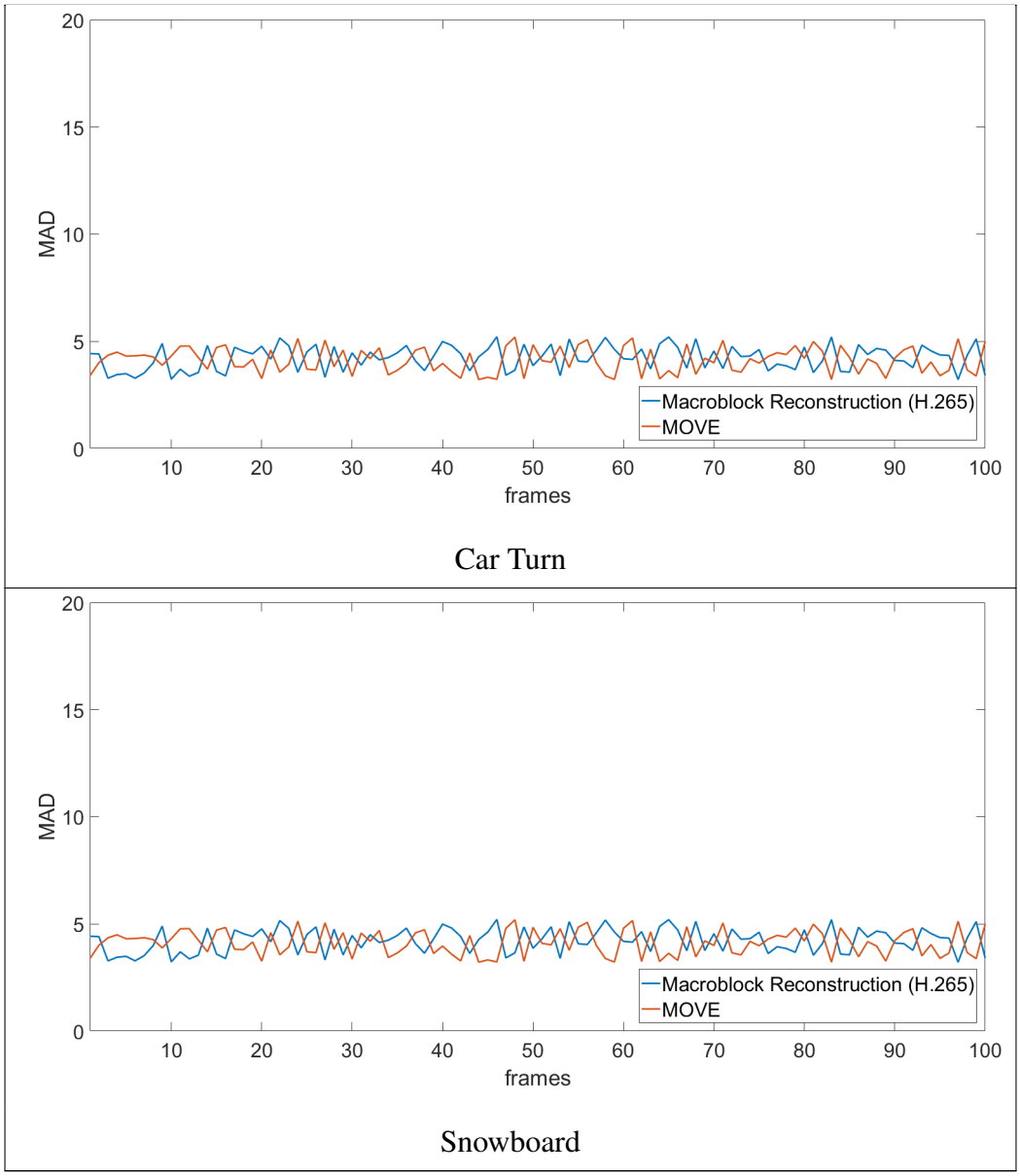
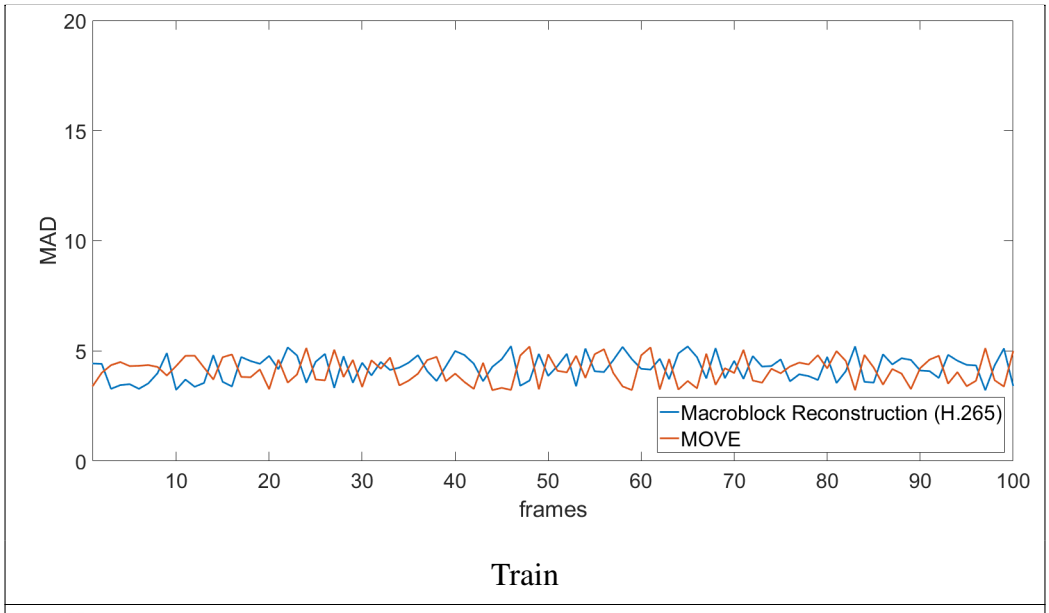
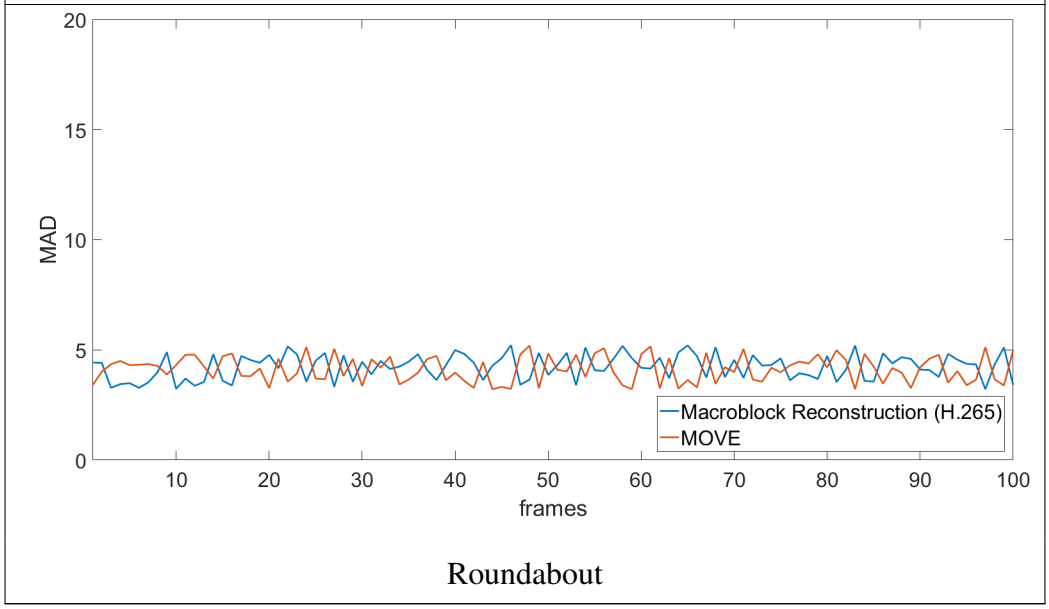


Table. 6.3 Continued on next page . . .



Train

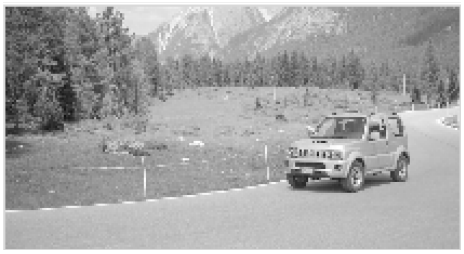
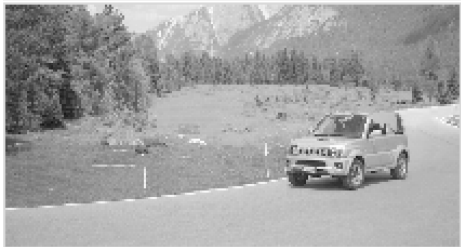
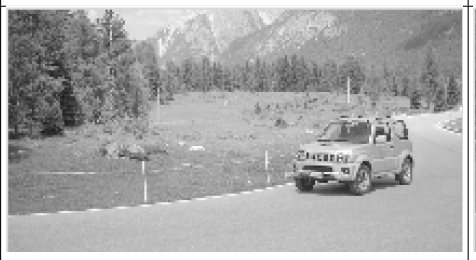


Roundabout

Table 6.4: Reconstructed frames using the macroblock based technique (H.265), a set of smooth layers generated by the model in Section. 3.1 and the proposed method (MOVE)

		Initial Frame	Final Frame
surf	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9617 **PSNR** = 26.77 dB **MAD** = 5.74	**SSIM** = 0.9618 **PSNR** = 26.78 dB **MAD** = 5.65
	Smooth Layers		
		SSIM = 0.9570 **PSNR** = 26.64 dB **MAD** = 6.29	**SSIM** = 0.9624 **PSNR** = 27.24 dB **MAD** = 5.83
	MOVE		
		SSIM = 0.9495 **PSNR** = 27.86 dB **MAD** = 5.15	**SSIM** = 0.9630 **PSNR** = 29.81 dB **MAD** = 3.66

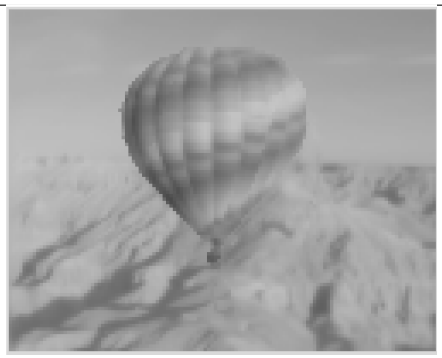
Table. 6.4 Continued on subsequent pages ...

car-turn	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9321 **PSNR** = 23.93 dB **MAD** = 9.91	**SSIM** = 0.9346 **PSNR** = 24.02 dB **MAD** = 9.77
	Smooth Layers		
		SSIM = 0.9549 **PSNR** = 26.02 dB **MAD** = 8.01	**SSIM** = 0.9335 **PSNR** = 24.23 dB **MAD** = 9.76
	MOVE		
		SSIM = 0.9660 **PSNR** = 31.58 dB **MAD** = 4.05	**SSIM** = 0.9373 **PSNR** = 27.15 dB **MAD** = 7.68

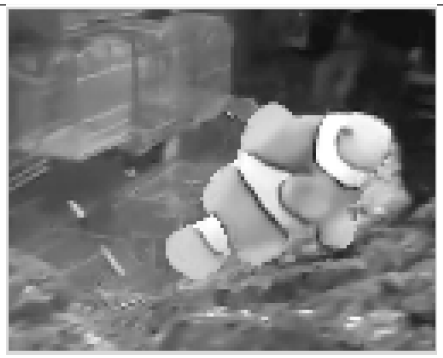
snowboard	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9458 **PSNR** = 21.57 dB **MAD** = 12.61	**SSIM** = 0.9528 **PSNR** = 22.37 dB **MAD** = 10.80
	Smooth Layers		
		SSIM = 0.9497 **PSNR** = 22.36 dB **MAD** = 11.76	**SSIM** = 0.9330 **PSNR** = 21.07 dB **MAD** = 13.42
	MOVE		
		SSIM = 0.9577 **PSNR** = 25.05 dB **MAD** = 6.41	**SSIM** = 0.9320 **PSNR** = 22.34 dB **MAD** = 10.62

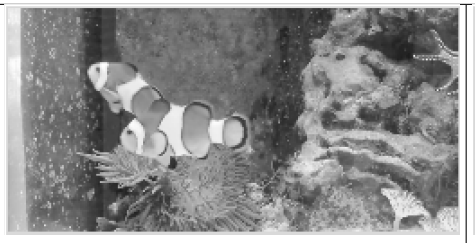
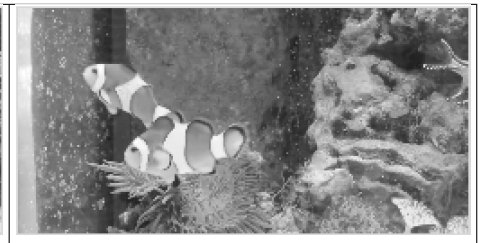
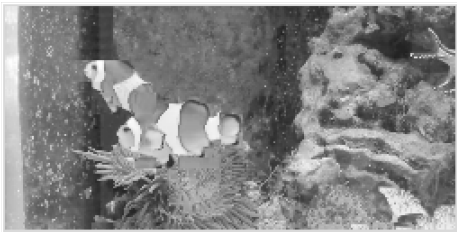
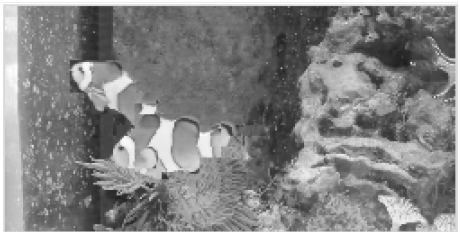
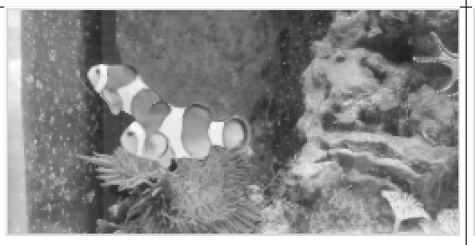
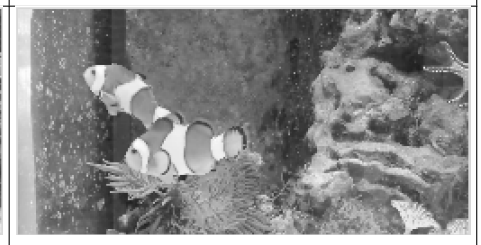
Train	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9279 **PSNR** = 23.71 dB **MAD** = 10.40	**SSIM** = 0.9293 **PSNR** = 23.80 dB **MAD** = 10.31
	Smooth Layers		
		SSIM = 0.9026 **PSNR** = 22.94 dB **MAD** = 11.98	**SSIM** = 0.9117 **PSNR** = 23.36 dB **MAD** = 11.45
	MOVE		
		SSIM = 0.9273 **PSNR** = 24.21 dB **MAD** = 10.83	**SSIM** = 0.9296 **PSNR** = 25.80 dB **MAD** = 8.63

roundabout	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9393 **PSNR** = 21.95 dB **MAD** = 10.32	**SSIM** = 0.9460 **PSNR** = 22.50 dB **MAD** = 9.80
	Smooth Layers		
		SSIM = 0.9520 **PSNR** = 23.53 dB **MAD** = 9.98	**SSIM** = 0.9340 **PSNR** = 22.13 dB **MAD** = 11.55
	MOVE		
		SSIM = 0.9649 **PSNR** = 26.67 dB **MAD** = 4.64	**SSIM** = 0.9396 **PSNR** = 23.56 dB **MAD** = 8.40

Hot Air Balloon	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9833 **PSNR** = 34.44 dB **MAD** = 2.57	**SSIM** = 0.9851 **PSNR** = 35.18 dB **MAD** = 2.46
	Smooth Layers		
		SSIM = 0.9742 **PSNR** = 32.82 dB **MAD** = 3.61	**SSIM** = 0.9708 **PSNR** = 32.53 dB **MAD** = 3.60
	MOVE		
		SSIM = 0.9849 **PSNR** = 36.46 dB **MAD** = 1.51	**SSIM** = 0.9819 **PSNR** = 35.52 dB **MAD** = 1.87

Ice Cream Van	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9515 **PSNR** = 28.51 dB **MAD** = 7.8	**SSIM** = 0.9389 **PSNR** = 22.99 dB **MAD** = 9.57
	Smooth Layers		
		SSIM = 0.9331 **PSNR** = 23.27 dB **MAD** = 9.52	**SSIM** = 0.9223 **PSNR** = 22.51 dB **MAD** = 10.35
	MOVE		
		SSIM = 0.9577 **PSNR** = 26.95 dB **MAD** = 4.65	**SSIM** = 0.9436 **PSNR** = 25.38 dB **MAD** = 6.31

Clownfish1	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9793 **PSNR** = 27.66 dB **MAD** = 5.78	**SSIM** = 0.9803 **PSNR** = 27.90 dB **MAD** = 5.72
	Smooth Layers		
		SSIM = 0.9625 **PSNR** = 25.35 dB **MAD** = 7.94	**SSIM** = 0.9448 **PSNR** = 23.63 dB **MAD** = 9.40
	MOVE		
		SSIM = 0.9696 **PSNR** = 28.00 dB **MAD** = 4.21	**SSIM** = 0.9437 **PSNR** = 24.59 dB **MAD** = 7.92

Clownfish2	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9561 **PSNR** = 24.78 dB **MAD** = 8.23	**SSIM** = 0.9571 **PSNR** = 24.95 dB **MAD** = 8.04
	Smooth Layers		
		SSIM = 0.9620 **PSNR** = 25.81 dB **MAD** = 8.31	**SSIM** = 0.9407 **PSNR** = 23.89 dB **MAD** = 10.20
	MOVE		
		SSIM = 0.9688 **PSNR** = 30.94 dB **MAD** = 4.28	**SSIM** = 0.9507 **PSNR** = 27.57 dB **MAD** = 6.82

Motorbikes	Original Frames		
	Macroblocks (H.265)		
		SSIM = 0.9385 **PSNR** = 22.96 dB **MAD** = 8.54	**SSIM** = 0.9579 **PSNR** = 24.41 dB **MAD** = 7.27
	Smooth Layers		
		SSIM = 0.9626 **PSNR** = 25.47 dB **MAD** = 7.28	**SSIM** = 0.9571 **PSNR** = 24.82 dB **MAD** = 7.83
	MOVE		
		SSIM = 0.9856 **PSNR** = 32.13 dB **MAD** = 2.01	**SSIM** = 0.9567 **PSNR** = 25.60 dB **MAD** = 6.41

Appendices

APPENDIX A

DERIVATION OF EQUATIONS IN SECTION 4

Eq. 2.6 and Eq. 2.7 are the Euler-Lagrange equations satisfied by f_{top} and f_{bottom} respectively of the functional $E_{appearance}$ Eq. 2.4 where f_{top} has been replaced with the solution f^*_{top} and $fbottom$ has been replaced with the solution f_{bottom} These are readily obtained by applying the (well known) Euler-Lagrange formula to Eq. 2.5.

Eq. 2.8 has been once again presented here.

$$
\begin{aligned}
E^*_{shape} &= \beta \int_{\partial R} ds + \int_R \lambda_{bottom} \Big(\Delta f^*_{bottom} \Big) dx \\
&+ \big(1-\alpha\big) \left(\int_R \big(I - f^*_{top}\big)^2 dx + \int_{\Omega \setminus R} \big(I - f^*_{bottom}\big)^2 dx \right) \\
&+ \int_R \lambda_{top} \Big(\alpha \Delta f^*_{top} + \big(1-\alpha\big)\big(I - f^*_{top}\big) \Big) dx \\
&+ \int_{\Omega \setminus R} \lambda_{bottom} \Big(\alpha \Delta f^*_{bottom} + \big(1-\alpha\big)\big(I - f^*_{bottom}\big) \Big) dx
\end{aligned}
\tag{2.8}
$$

We can rewrite Eq. 2.8 as

$$
E^*_{shape} = E_{shapeprior} + E_{top,constrained} + E_{bottom,constrained} \tag{A 1}
$$

where

$$E_{shapeprior} = \beta \int_{\partial R} ds \tag{A 2}$$

$$E_{top,constrained} = \int_R \big(1-\alpha\big)\big(I - f^*_{top}\big)^2 dR + \int_R \lambda_{top}(\alpha \Delta f^*_{top} + \big(1-\alpha\big)\big(I - f^*_{top}\big))dx \tag{A 3}$$

$$\begin{aligned} E_{bottom,constrained} = & \int_{\Omega \backslash R} \big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 dx \\ & + \int_{\Omega \backslash R} \lambda_{bottom}(\alpha \Delta f^*_{bottom} + \big(1-\alpha\big)\big(I - f^*_{bottom}\big))dx \\ & + \int_R \lambda_{bottom} \alpha \Delta f^*_{bottom} dx \end{aligned} \tag{A 4}$$

Rearranging we have

$$E_{bottom,constrained} = \int_{\Omega \setminus R} \big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \lambda_{bottom}\big(1-\alpha\big)\big(I - f^*_{bottom}\big)dx + \int_{\Omega} \lambda_{bottom}\alpha\Delta f^*_{bottom}dx \tag{A 5}$$

From integration by parts we have

$$\int_R \alpha\lambda_{top}\Delta f^*_{top}dx = \int_C \alpha\lambda_{top}\nabla f^*_{top}\bullet N ds - \int_R \alpha\nabla\lambda_{top}\bullet\nabla f^*_{top}dx \tag{A 6}$$

Substituting Eq. A 6 into Eq. A 3 we have

$$E_{top,constrained} = \int_R \big(1-\alpha\big)\big(I - f^*_{top}\big)^2 dx - \int_R \Big(\alpha\nabla\lambda_{top}\bullet\nabla f^*_{top} - \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big)\Big)dx + \underbrace{\int_C \alpha\lambda_{top}\nabla f^*_{top}\bullet N ds}_{\text{Integral 1}} \tag{A 7}$$

Integral 1 can be driven to zero by imposing vanishing Neumann boundary conditions for

f^*_{top} over R. We now get

$$E_{top,constrained} = \int_R \Big(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha \nabla \lambda_{top} \bullet \nabla f^*_{top} \Big) dx \tag{A 8}$$

To compute the gradient of $E_{top,constrained}$ with respect to the curve C we use Eq. A 8 and let the curve C vary with time. Now

$$\begin{aligned} &\frac{\partial E_{top,constrained}}{\partial t} = \\ &\int_C \Big(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha \nabla \lambda_{top} \bullet \nabla f^*_{top} \Big) \frac{\partial C}{\partial t} \bullet N ds \\ &+ \int_R \Big(-2\big(1-\alpha\big)\big(I - f^*_{top}\big)\frac{\partial f^*_{top}}{\partial t} - \big(1-\alpha\big)\lambda_{top}\frac{\partial f^*_{top}}{\partial t} - \alpha \nabla \lambda_{top} \bullet \nabla \frac{\partial f^*_{top}}{\partial t} \Big) dx \\ &+ \int_R \Big(\big(1-\alpha\big)\big(I - f^*_{top}\big)\frac{\partial \lambda_{top}}{\partial t} - \alpha \nabla f^*_{top} \bullet \nabla \frac{\partial \lambda_{top}}{\partial t} \Big) dx \end{aligned} \tag{A 9}$$

Using integration by parts we have

$$\int_R \Big(\alpha \nabla \lambda_{top} \bullet \nabla \frac{\partial f^*_{top}}{\partial t} \Big) dx = \int_C \alpha \nabla \lambda_{top} \bullet N \frac{\partial f^*_{top}}{\partial t} ds - \int_R \alpha \Delta \lambda_{top} \frac{\partial f^*_{top}}{\partial t} dx \tag{A 10}$$

$$\int_R \Big(\alpha \nabla f^*_{top} \bullet \nabla \frac{\partial \lambda_{top}}{\partial t} \Big) dx = \int_C \alpha \nabla f^*_{top} \bullet N \frac{\partial \lambda_{top}}{\partial t} ds - \int_R \alpha \Delta f^*_{top} \frac{\partial \lambda_{top}}{\partial t} dx \tag{A 11}$$

Substituting Eq. A 10 and Eq. A 11 into Eq. A 9 yeilds

$$\frac{\partial E_{top,constrained}}{\partial t} = \int_C \left(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha\nabla\lambda_{top}\bullet\nabla f^*_{top} \right) \frac{\partial C}{\partial t}\bullet N ds$$
$$+ \underbrace{\int_R \left(-2\big(1-\alpha\big)\big(I - f^*_{top}\big) - \big(1-\alpha\big)\lambda_{top} + \alpha\Delta\lambda_{top} \right) \frac{\partial f^*_{top}}{\partial t} dx}_{\text{Integral 2}} - \underbrace{\int_C \alpha\nabla\lambda_{top}\bullet N \frac{\partial f^*_{top}}{\partial t} ds}_{\text{Integral 3}}$$
$$+ \underbrace{\int_R \left(\big(1-\alpha\big)\big(I - f^*_{top}\big) + \alpha\Delta f^*_{top} \right) \frac{\partial \lambda_{top}}{\partial t} dx}_{\text{Integral 4}} - \underbrace{\int_C \alpha\nabla f^*_{top}\bullet N \frac{\partial \lambda_{top}}{\partial t} ds}_{\text{Integral 5}} \tag{A 12}$$

Integral 2 can be forced to zero by imposing Eq. 2.10 and Integral 3 can be forced to zero by imposing vanishing Neumann boundary conditions for λ_{top} over R. Integral 4 is zero by Eq. 2.6 and Integral 5 is zero by the vanishing Neumann boundary conditions imposed for f^*_{top} earlier. Eq. A 12 therefore yields the gradient of $E_{top,constrained}$ with respect to the curve C as

$$\nabla_C E_{top,constrained} = \left(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha\nabla\lambda_{top}\bullet\nabla f^*_{top} \right) N \tag{A 13}$$

Let $\partial\Omega$ be the boundary of Ω, N_Ω be the outward unit normal to Ω at any particular

point ds_Ω be an infinitesimally small distance along $\partial\Omega$. From integration by parts we have

$$\int_\Omega \alpha\lambda_{bottom}\Delta f^*_{bottom}dx = \int_{\partial\Omega} \alpha\lambda_{bottom}\nabla f^*_{bottom}\bullet N_\Omega ds_\Omega - \int_\Omega \alpha\nabla\lambda_{bottom}\bullet\nabla f^*_{bottom}dx \tag{A 14}$$

Substituting Eq. A 14 into Eq. A 5 we have

$$E_{bottom,constrained} = \int_{\Omega\setminus R}\Big(\big(1-\alpha\big)\big(I-f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I-f^*_{bottom}\big)\Big)dx$$
$$\underbrace{+\int_{\partial\Omega}\alpha\lambda_{bottom}\nabla f^*_{bottom}\bullet N_\Omega ds_\Omega}_{\text{Integral 6}} - \int_\Omega \alpha\nabla\lambda_{bottom}\bullet\nabla f^*_{bottom} \tag{A 15}$$

Integral 6 can be made 0 by imposing vanishing Neumann boundary conditions on f^*_{bottom} over Ω. To compute the gradient of $E_{bottom,constrained}$ with respect to the curve C we use

Eq. A 15 and let the curve C vary with time. Now

$$\begin{aligned}
&\frac{\partial E_{bottom,constrained}}{\partial t} = \\
&\int_C \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) \right) \frac{\partial C}{\partial t} \bullet \big(-N\big) ds \\
&+ \int_{\Omega \setminus R} \left(-2\big(1-\alpha\big)\big(I - f^*_{bottom}\big) - \big(1-\alpha\big)\lambda_{bottom} \right) \frac{\partial f^*_{bottom}}{\partial t} dx \\
&+ \int_{\Omega \setminus R} \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big) \right) \frac{\partial \lambda_{bottom}}{\partial t} dx \\
&- \int_{\Omega} \left(\alpha \nabla \lambda_{bottom} \bullet \nabla \frac{\partial f^*_{bottom}}{\partial t} \right) dx \\
&- \int_{\Omega} \left(\alpha \nabla f^*_{bottom} \bullet \nabla \frac{\partial \lambda_{bottom}}{\partial t} \right) dx
\end{aligned} \tag{A 16}$$

Using integration by parts we have

$$\begin{aligned}
&\int_{\Omega} \left(\alpha \nabla \lambda_{bottom} \bullet \nabla \frac{\partial f^*_{bottom}}{\partial t} \right) dx = \\
&\int_{\partial\Omega} \alpha \nabla \lambda_{bottom} \bullet N_{\Omega} \frac{\partial f^*_{bottom}}{\partial t} ds_{\Omega} - \int_{\Omega} \alpha \Delta \lambda_{bottom} \frac{\partial f^*_{bottom}}{\partial t} dx
\end{aligned} \tag{A 17}$$

$$\begin{aligned}
&\int_{\Omega} \left(\alpha \nabla f^*_{bottom} \bullet \nabla \frac{\partial \lambda_{bottom}}{\partial t} \right) dx = \\
&\int_{\partial\Omega} \alpha \nabla f^*_{bottom} \bullet N_{\Omega} \frac{\partial \lambda_{bottom}}{\partial t} ds_{\Omega} - \int_{\Omega} \alpha \Delta f^*_{bottom} \frac{\partial \lambda_{bottom}}{\partial t} dx
\end{aligned} \tag{A 18}$$

Substituting Eq. A 17 and Eq. A 18 into Eq. A 16 yeilds

$$
\begin{aligned}
&\int_C \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) \right) \frac{\partial C}{\partial t} \bullet \big(-N\big) ds \\
&+ \int_{\Omega \backslash R} \left(-2\big(1-\alpha\big)\big(I - f^*_{bottom}\big) - \big(1-\alpha\big)\lambda_{bottom} \right) \frac{\partial f^*_{bottom}}{\partial t} dx \\
&+ \int_{\Omega \backslash R} \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big) \right) \frac{\partial \lambda_{bottom}}{\partial t} dx \\
&- \int_{\partial\Omega} \alpha \nabla \lambda_{bottom} \bullet N_\Omega \frac{\partial f^*_{bottom}}{\partial t} ds_\Omega + \int_\Omega \alpha \Delta \lambda_{bottom} \frac{\partial f^*_{bottom}}{\partial t} dx \\
&- \int_{\partial\Omega} \alpha \nabla f^*_{bottom} \bullet N_\Omega \frac{\partial \lambda_{bottom}}{\partial t} ds_\Omega + \int_\Omega \alpha \Delta f^*_{bottom} \frac{\partial \lambda_{bottom}}{\partial t} dx
\end{aligned}
\tag{A 19}
$$

which simplifies to

$$
\begin{aligned}
&\int_C \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) \right) \frac{\partial C}{\partial t} \bullet \big(-N\big) ds \\
&+ \underbrace{\int_{\Omega \backslash R} \left(-2\big(1-\alpha\big)\big(I - f^*_{bottom}\big) - \big(1-\alpha\big)\lambda_{bottom} + \alpha\Delta\lambda_{bottom} \right) \frac{\partial f^*_{bottom}}{\partial t} dx}_{\text{Integral 7}} \\
&+ \underbrace{\int_{\Omega \backslash R} \left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big) + \alpha\Delta f^*_{bottom} \right) \frac{\partial \lambda_{bottom}}{\partial t} dx}_{\text{Integral 8}} \\
&- \underbrace{\int_{\partial\Omega} \alpha \nabla \lambda_{bottom} \bullet N_\Omega \frac{\partial f^*_{bottom}}{\partial t} ds_\Omega}_{\text{Integral 9}} + \underbrace{\int_R \alpha \Delta \lambda_{bottom} \frac{\partial f^*_{bottom}}{\partial t} dx}_{\text{Integral 10}} \\
&- \underbrace{\int_{\partial\Omega} \alpha \nabla f^*_{bottom} \bullet N_\Omega \frac{\partial \lambda_{bottom}}{\partial t} ds_\Omega}_{\text{Integral 11}} + \underbrace{\int_R \alpha \Delta f^*_{bottom} \frac{\partial \lambda_{bottom}}{\partial t} dx}_{\text{Integral 12}}
\end{aligned}
\tag{A 20}
$$

Integral 7 and Integral 10 can be made zero by imposing Eq. 2.11. Integral 8 and Integral 12 are zero by Eq. 2.7. Integral 11 is zero by the Neumann boundary condition imposed earlier for f^*_{bottom} and Integral 9 can be made zero by imposing vanishing Neumann boundary conditions on λ_{bottom} over Ω. We therefore have from Eq. A 20 the gradient of $E_{bottom,constrained}$ with respect to the curve C as

$$\nabla_C E_{bottom,constrained} = -\left(\big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) \right) N \tag{A 21}$$

The gradient of an energy with respect to a curve such that the energy penalizes the arc length of the curve (when evaluated at a particular point on the curve) is known to be the outward unit normal wighted by the mean curvature at that point. Therefore from Eq. A 2 we have

$$\nabla_C E_{shapeprior} = \beta\kappa N \tag{A 22}$$

where κ is the mean curvature of C.

From Eq. A 1 we have

$$\nabla_C E_{shape,constrained} = \nabla_C E_{shapeprior} + \nabla_C E_{top,constrained} + \nabla_C E_{bottom,constrained} \tag{A 23}$$

Substituting Eq. A 13, Eq. A 21 and Eq. A 22 into Eq. A 23 yields

$$\begin{aligned}
&\nabla_C E^*_{shape} = \\
&\Bigg(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha\nabla\lambda_{top}\bullet\nabla f^*_{top} \Bigg) N \\
&- \Bigg(\big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 + \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) \Bigg) N \\
&+ \beta\kappa N
\end{aligned} \tag{A 24}$$

From Eq. A 24 we get Eq. 2.9 (the gradient descent PDE) for optimizing Eq. 2.8 with respect to the curve as

$$\begin{aligned}
&\frac{\partial C}{\partial t} = -\nabla_C E^*_{shape} \\
&= -\Bigg(\big(1-\alpha\big)\big(I - f^*_{top}\big)^2 + \big(1-\alpha\big)\lambda_{top}\big(I - f^*_{top}\big) - \alpha\nabla\lambda_{top}\bullet\nabla f^*_{top} \cdots \\
&\qquad - \big(1-\alpha\big)\big(I - f^*_{bottom}\big)^2 - \big(1-\alpha\big)\lambda_{bottom}\big(I - f^*_{bottom}\big) + \beta\kappa \Bigg) N
\end{aligned} \tag{2.9}$$

APPENDIX B

DERIVATION OF EQUATIONS IN SECTION 5

B.1 Preliminary work

We will use the following relationships in the derivations.

$$\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}} \circ g_{l,n} = -\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right] \tag{A 25}$$

Proof

Since $g_{l,n}$ is an invertible mapping from R_l to Ω_n and depends on the parameter $p_{l,n,k}$ for each point x acted on by $g_{l,n}$ we can write

$g_{l,n}^{-1}(g_{l,n}(x, p_{l,n,k}), p_{l,n,k}) = x$

Therfore

$$\frac{\partial g_{l,n}^{-1}}{\partial p_{l,n,k}}(g_{l,n}(x, p_{l,n,k}), p_{l,n,k}) = 0$$

$$\frac{\partial g_{l,n}^{-1}}{\partial p_{l,n,k}} \circ g_{l,n} + \left(\left[\text{Jacobian}\left(g_{l,n}^{-1}\right)\right] \circ g_{l,n}\right)\frac{\partial g_{l,n}}{\partial p_{l,n,k}}$$

Since $g_{l,n}$ is invertible its Jacobian is also invertible and we can apply the inverse function theorem

$$= \left[\text{Jacobian}\left(g_{l,n}^{-1}\right)\right] \circ g_{l,n} = \left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-1}$$

Substituting this result yields

$$\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}} \circ g_{l,n} = -\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]$$

$$\widehat{N}_{l,n}\widehat{ds}_{l,n} = \det\left(\text{Jacobian}\left(g_{l,n}\right)\right)\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-T} N_l ds_l \tag{A 26}$$

Proof

Let T_l and $\widehat{T}_{l,n}$ be the unit tangents to the curves C_l and $\widehat{C}_{l,n}$ respectively that are parameterized with respect to some parameter p and let J be the 2x2 rotation by 90° matrix such that

$N_l = JT_l$ and $\widehat{N}_{l,n} = J\widehat{T}_{l,n}$

Therefore

$$\widehat{N}_{l,n}\widehat{ds}_{l,n} = J\widehat{T}_{l,n}\widehat{ds}_{l,n} = J\frac{\partial\widehat{C}_{l,n}/\partial p}{\left\|\partial\widehat{C}_{l,n}/\partial p\right\|}\left\|\partial\widehat{C}_{l,n}/\partial p\right\| dp$$

$$= J\frac{\partial}{\partial p}\big(g_{l,n}\big(C_l\big)\big)dp$$

$$= J\big[\text{Jacobian}\left(g_{l,n}\right)\big]\frac{\partial C_l}{\partial p}dp$$

$$= J\big[\text{Jacobian}\left(g_{l,n}\right)\big]\frac{\partial C_l/\partial p}{\|\partial C_l/\partial p\|}\|\partial C_l/\partial p\|\, dp$$

$$= J\big[\text{Jacobian}\left(g_{l,n}\right)\big]T_l ds_l$$

$$= J\big[\text{Jacobian}\left(g_{l,n}\right)\big]J^T N_l ds_l \text{ (using } J^T N = J^T JT \text{ where } J^T J = I\text{)}$$

Note that for any 2x2 matrix A, the corresponding cofactor matrix, or equivalently the transpose of the adjugate, can be expressed as JAJ^T

Therefore

$$\widehat{N}_{l,n}\widehat{ds}_{l,n} = \big[\text{adj}\big(\text{Jacobian}\left(g_{l,n}\right)\big)\big]^T N_l ds_l$$

which implies that

$$\widehat{N}_{l,n}\widehat{ds}_{l,n} = \det\left(\text{Jacobian}\left(g_{l,n}\right)\right)\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-T} N_l ds_l$$

$$\frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{ds}_{l,n} = \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) \frac{\partial C_l}{\partial t} \bullet N_l ds_l \tag{A 27}$$

Proof

$$\frac{\partial \widehat{C}_{l,n}}{\partial t} = \frac{\partial}{\partial t} g_{l,n}(C_l) = \left[\text{Jacobian}\left(g_{l,n}\right)\right]\left[\frac{\partial C_l}{\partial t}\right]$$

Using this result and Eq.A26 we have

$$\frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{ds}_{l,n} =$$

$$\left[\text{Jacobian}\left(g_{l,n}\right)\right]\left[\frac{\partial C_l}{\partial t}\right] \bullet \det\left(\text{Jacobian}\left(g_{l,n}\right)\right)\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-T} N_l ds_l$$

$$= \det\left(\text{Jacobian}\left(g_{l,n}\right)\right)\left[\frac{\partial C_l}{\partial t}\right]^{T}\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{T}\left[\text{Jacobian}\left(g_{l,n}\right)\right]^{-T} N_l ds_l$$

$$= \det\left(\text{Jacobian}\left(g_{l,n}\right)\right)\left[\frac{\partial C_l}{\partial t}\right] \bullet N_l ds_l$$

Therefore

$$\frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{ds}_{l,n} = \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) \frac{\partial C_l}{\partial t} \bullet N_l ds_l$$

$$\int_{\widehat{C}_{l,n}} f(x) \frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{ds}_{l,n} = \int_{C_l} \left(f \circ g_{l,n}(x)\right) \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) \frac{\partial C_l}{\partial t} \bullet N_l ds_l \tag{A 28}$$

Proof

making the substitution $u = g_{l,n}^{-1}(x)$ and using Eq.A27 we have

$$\int_{\widehat{C}_{l,n}} f(x) \frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{d}s_{l,n} = \int_{C_l} (f \circ g_{l,n}(u)) \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) \frac{\partial C_l}{\partial t} \bullet N_l ds_l$$

making the substitution $x = u$ we have

$$\int_{\widehat{C}_{l,n}} f(x) \frac{\partial \widehat{C}_{l,n}}{\partial t} \bullet \widehat{N}_{l,n} \widehat{d}s_{l,n} = \int_{C_l} (f \circ g_{l,n}(x)) \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) \frac{\partial C_l}{\partial t} \bullet N_l ds_l$$

$$\int_{\Omega_n} \delta_{l,visible_n} f(x) dx = \int_{g_{l,n}\left(R_l\right)} \delta_{l,visible_n} f(x) dx \tag{A 29}$$

Proof

$\delta_{l,visible_n}$ is only true over a subset of $g_{l,n}\left(R_l\right)$ and therefore this relationship holds

$$\int_{\Omega_n} \delta_{l,visible_n} f(x) dx = \int_{R_l} V_{l,n} (f \circ g_{l,n}(x)) dx \tag{A 30}$$

Proof

Using Eq. A 29

$$\int_{\Omega_n} \delta_{l,visible_n} f(x)dx = \int_{g_{l,n}\left(R_l\right)} \delta_{l,visible_n} f(x)dx$$

making the substitution $u = g_{l,n}^{-1}(x)$ we have

$$= \int_{R_l} (\delta_{l,visible_n \circ g_{l,n}}) \det\left(\text{Jacobian}\left(g_{l,n}\right)\right) (f \circ g_{l,n}(u))du$$

making the substitution $x = u$ we have

$$= \int_{R_l} V_{l,n}(f \circ g_{l,n}(x))dx$$

We also introduce the following notation

$$\begin{aligned}
&\tilde{f}_{l,n}^{*} = f_l^{*} \circ g_{l,n}^{-1} \\
&\tilde{\lambda}_{l,n} = \left(\lambda_l \circ g_{l,n}^{-1}\right) \\
&\widehat{C}_{l,n} = g_{l,n}\left(C_l\right) \\
&\overline{f}_{l,n}(x) = f_{next_n \circ g_{l,n}(x)} \circ g^{-1}_{next_n \circ g_{l,n}(x),n} \circ g_{l,n}(x)
\end{aligned} \tag{A 31}$$

B.2 Derivation of results

From Eq. 3.2 we can write

$$\int_{\Omega_n} \left(I_n - f_{visible_n} \circ g^{-1}_{visible_n,n}\right)^2 dx = \int_{\Omega_n} \sum_{l=1}^{L} \delta_{l,visible_n} \left(I_n - f_l \circ g_{l,n}^{-1}\right)^2 dx \tag{A 32}$$

Interchanging the order of summation and integration we can write

$$\int_{\Omega_n} \sum_{l=1}^{L} \delta_{l,visible_n} \big(I_n - f_l \circ g_{l,n}^{-1}\big)^2 dx = \sum_{l=1}^{L} \int_{\Omega_n} \delta_{l,visible_n} \big(I_n - f_l \circ g_{l,n}^{-1}\big)^2 dx$$

Using Eq. A 30 we get

$$\sum_{l=1}^{L} \int_{\Omega_n} \delta_{l,visible_n} \big(I_n - f_l \circ g_{l,n}^{-1}\big)^2 dx = \sum_{l=1}^{L} \int_{R_l} V_{l,n} \big(I_n \circ g_{l,n} - f_l\big)^2 dx$$

which therefore gives

$$\int_{\Omega_n} \big(I_n - f_{visible_n} \circ g_{visible_n,n}^{-1}\big)^2 dx = \sum_{l=1}^{L} \int_{R_l} V_{l,n} \big(I_n \circ g_{l,n} - f_l\big)^2 dx \tag{A 33}$$

Substituting Eq. A 33 into the appearance energy (Eq. 3.2) we get

$$E_{appearance} = \sum_{l=1}^{L} \alpha \int_{R_l} \|\nabla f_l\|^2 dR_l + \big(1-\alpha\big) \sum_{l=1}^{L} \sum_{n=1}^{N} \gamma_n \int_{R_l} V_{l,n} \big(I_n \circ g_{l,n} - f_l\big)^2 dx \tag{A 34}$$

To optimize the appearance energy Eq. A 34 with respect to an appearance function f_k where $k \in [1\ L]$, the function f_k needs to satisfy the Euler-Lagrange equation of Eq. A 34 with respect to f_k over the domain R_k given by

$$\big(1-\alpha\big) \sum_{n=1}^{N} \Big(\gamma_n V_{k,n} \big(I_n \circ g_{k,n} - f_k\big) \Big) + \alpha \Delta f_k = 0 \tag{A 35}$$

Let $f_k^*\ k \in [1\ L]$ be the set of optimizers for the smooth appearances of the layers over the regions $R_k\ k \in [1\ L]$. We therefore have from Eq. A 35

$$\big(1-\alpha\big)\sum_{n=1}^{N}\left(\gamma_n V_{k,n}\big(D_{k,n}^*\big)\right)+\alpha\Delta f_k^*=0, \qquad k\in[1\ L], \qquad \text{over } R_k \tag{3.4}$$

which is essentially Eq. 3.4. The issue of Nuemann boundary conditions in Eq. 3.4 will be discussed later. Let $\lambda_l\ l \in [1\ L]$ be a set of pointwise Lagrange multipliers to impose the constraints given in Eq. 3.4. Substituting Eq. A 33 along with the constraints of Eq. 3.4 into the shape energy Eq. 3.1 we get a constrained shape energy as

$$\begin{aligned}
E_{shape}^* &= \sum_{l=1}^{L}\beta_l\int_{\partial R_l} ds_l \\
&+ \sum_{n=1}^{N}\gamma_n\big(1-\alpha\big)\int_{\Omega_n}\big(I_n - f^*_{visible_n}\circ g^{-1}_{visible_n,n}\big)^2 dx \\
&+ \sum_{l=1}^{L}\int_{R_l}\lambda_l\left(\sum_{n=1}^{N}\big(1-\alpha\big)\left(\gamma_n V_{l,n}\big(\big(I_n\circ g_{l,n}-f_l^*\big)\big)\right)+\alpha\Delta f_l^*\right)dx
\end{aligned} \tag{A 36}$$

making the substitution $u = g_{l,n}(x)$ and using Eq. A 32 in Eq. A 36 we get

$$
\begin{aligned}
&= \sum_{l=1}^{L} \beta_l \int_{C_l} ds_l \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} \big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - f_l^* \circ g_{l,n}^{-1}\big)^2 dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} \big(\lambda_l \circ g_{l,n}^{-1}\big)\big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - f_l^* \circ g_{l,n}^{-1}\big) du \\
&+ \sum_{l=1}^{L} \int_{R_l} \lambda_l \alpha \Delta f_l^* dx
\end{aligned}
\tag{A 37}
$$

making the substitution $x = u$ and using the notation of Eq. A 31 followed by integration by parts on the integral over R_l we get

$$
\begin{aligned}
E^*_{shape} &= \sum_{l=1}^{L} \beta_l \int_{\partial R_l} ds_l \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} \big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - \tilde{f}^*_{l,n}\big)^2 dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} \tilde{\lambda}_{l,n}\big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - \tilde{f}^*_{l,n}\big) dx \\
&+ \sum_{l=1}^{L} \Bigg(\underbrace{\int_{C_l} \alpha \lambda_l \nabla f_l^* \bullet N_l ds_l}_{\text{Integral B1}} - \int_{R_l} \alpha \nabla \lambda_l \bullet \nabla f_l^* dx \Bigg)
\end{aligned}
\tag{A 38}
$$

Integral B1 in Eq. A 38 can be set to zero by enforcing Neumann boundary conditions on f_l over the region R_l. Let C_k be the boundary of R_k and be allowed to vary with time. We therefore have

$$\frac{\partial}{\partial t}E^*_{shape} = \beta_k \int_{C_k} \kappa_k \frac{\partial C_k}{\partial t} \cdot N_k ds_k - \int_{C_k} \alpha \nabla \lambda_k \cdot \nabla f^*_k \frac{\partial C_k}{\partial t} \cdot N_k ds_k$$
$$+ \sum_{n=1}^{N} \gamma_n \int_{\widehat{C}_{k,n}} \big(1-\alpha\big)\delta_{k,visible_n} \ldots$$
$$\left(\big(I_n - \tilde{f}^*_{k,n}\big)^2 + \tilde{\lambda}_{k,n}\big(I_n - \tilde{f}^*_{k,n}\big) - \big(I_n - \tilde{f}^*_{next_n,n}\big)^2 - \tilde{\lambda}_{next_n,n}\big(I_n - \tilde{f}^*_{next_n,n}\big) \right) \frac{\partial \widehat{C}_{k,n}}{\partial t} \cdot \widehat{N}_{k,n} \widehat{ds}_{k,n}$$
$$+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} -2\big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - \tilde{f}^*_{l,n}\big)\frac{\partial \tilde{f}^*_{l,n}}{\partial t} dx$$
$$+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} -\big(1-\alpha\big)\tilde{\lambda}_{l,n}\delta_{l,visible_n}\frac{\partial \tilde{f}^*_{l,n}}{\partial t} dx - \sum_{l=1}^{L} \int_{R_l} \alpha \nabla \lambda_l \cdot \nabla \frac{\partial f^*_l}{\partial t}$$
$$+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{\Omega_n} \big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - \tilde{f}^*_{l,n}\big)\frac{\partial \tilde{\lambda}_{l,n}}{\partial t} dx - \sum_{l=1}^{L} \int_{R_l} \alpha \nabla f^*_l \cdot \nabla \frac{\partial \lambda_l}{\partial t}$$

Using Eq. A 28 and integration by parts on the integrals over R_l

$$
\begin{aligned}
&= \beta_k \int_{C_k} \kappa_k \frac{\partial C_k}{\partial t} \bullet N_k ds_k - \int_{C_k} \alpha \nabla \lambda_k \bullet \nabla f_k \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{n=1}^{N} \gamma_n \int_{C_k} \big(1-\alpha\big) V_{k,n} \dots \\
&\left(\big(I_n \circ g_{k,n} - f_k^*\big)^2 + \lambda_k \big(I_n \circ g_{k,n} - f_k^*\big) - \big(I_n \circ g_{k,n} - \overline{f}_{k,n}^*\big)^2 - \overline{\lambda}_{k,n} \big(I_n \circ g_{k,n} - \overline{f}_{k,n}^*\big) \right) \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} -2\big(1-\alpha\big) \delta_{l,visible_n} \big(I_n - \tilde{f}_{l,n}\big) \frac{\partial \tilde{f}_{l,n}^*}{\partial t} dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} -\big(1-\alpha\big) \tilde{\lambda}_{l,n} \delta_{l,visible_n} \frac{\partial \tilde{f}_{l,n}^*}{\partial t} dx \\
&- \sum_{l=1}^{L} \Bigg(\underbrace{\int_{C_l} \alpha \nabla \lambda_l \bullet N_l \frac{\partial f_l^*}{\partial t}}_{\text{Integral B2}} + \int_{R_l} \alpha \Delta \lambda_l \frac{\partial f_l^*}{\partial t} \Bigg) \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} \big(1-\alpha\big) \delta_{l,visible_n} \big(I_n - \tilde{f}_{l,n}\big) \frac{\partial \tilde{\lambda}_{l,n}}{\partial t} dx \\
&- \sum_{l=1}^{L} \Bigg(\underbrace{\int_{C_l} \alpha \nabla f_l^* \bullet N_l \frac{\partial \lambda_l}{\partial t}}_{\text{Integral B3}} + \int_{R_l} \alpha \Delta f_l^* \frac{\partial \lambda_l}{\partial t} \Bigg)
\end{aligned}
$$

Integral B2 can be set to zero by enforcing Neumann boundary conditions on λ_l over the region R_l and Integral B3 is zero by the Nuemann boundary conditions imposed earlier.

We therefore have

$$
\begin{aligned}
&= \beta_k \int_{C_k} \kappa_k \frac{\partial C_k}{\partial t} \bullet N_k ds_k - \int_{C_k} \alpha \nabla \lambda_k \bullet \nabla f_k \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{n=1}^{N} \gamma_n \int_{C_k} \big(1-\alpha\big) V_{k,n} \left((D^*_{k,n})^2 + \lambda_k D^*_{k,n} - (\overline{D}^*_{k,n})^2 - \overline{\lambda}_{k,n} \overline{D}^*_{k,n} \right) \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} - 2\big(1-\alpha\big) V_{l,n} \big(I_n \circ g_{l,n} - f^*_l \big) \frac{\partial f^*_l}{\partial t} dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} - \big(1-\alpha\big) \lambda_l V_{l,n} \frac{\partial f^*_l}{\partial t} dx + \sum_{l=1}^{L} \int_{R_l} \alpha \Delta \lambda_l \frac{\partial f^*_l}{\partial t} \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} \big(1-\alpha\big) V_{l,n} \big(I_n \circ g_{l,n} - f^*_l \big) \frac{\partial \lambda_l}{\partial t} dx + \sum_{l=1}^{L} \int_{R_l} \alpha \Delta f^*_l \frac{\partial \lambda_l}{\partial t}
\end{aligned}
$$

$$
\begin{aligned}
&= \int_{C_k} \Bigg(\big(1-\alpha\big) \sum_{n=1}^{N} \gamma_n V_{k,n} \dots \\
&\left((D^*_{k,n})^2 + \lambda_k D^*_{k,n} - (\overline{D}^*_{k,n})^2 - \overline{\lambda}_{k,n} \overline{D}^*_{k,n} \right) - \nabla \lambda_k \bullet \nabla f_k + \beta_k \kappa_k \Bigg) \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} - 2\big(1-\alpha\big) V_{l,n} D^*_{l,n} \frac{\partial f^*_l}{\partial t} dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} - \big(1-\alpha\big) \lambda_l V_{l,n} \frac{\partial f^*_l}{\partial t} dx + \sum_{l=1}^{L} \int_{R_l} \alpha \Delta \lambda_l \frac{\partial f^*_l}{\partial t} \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{R_l} \big(1-\alpha\big) V_{l,n} D^*_{l,n} \frac{\partial \lambda_l}{\partial t} dx + \sum_{l=1}^{L} \int_{R_l} \alpha \Delta f^*_l \frac{\partial \lambda_l}{\partial t}
\end{aligned}
$$

$$
\begin{aligned}
&= \int_{C_k} \Bigg(\big(1-\alpha\big) \sum_{n=1}^{N} \gamma_n V_{k,n} \dots \\
&\left((D^*_{k,n})^2 + \lambda_k D^*_{k,n} - (\overline{D}^*_{k,n})^2 - \overline{\lambda}_{k,n} \overline{D}^*_{k,n} \right) - \nabla \lambda_k \bullet \nabla f_k + \beta_k \kappa_k \Bigg) \frac{\partial C_k}{\partial t} \bullet N_k ds_k \\
&+ \sum_{l=1}^{L} \int_{R_l} \left(\alpha \Delta \lambda_l - \lambda_l \big(1-\alpha\big) \left(\sum_{n=1}^{N} \gamma_n V_{l,n} \right) - 2\big(1-\alpha\big) \left(\sum_{n=1}^{N} \gamma_n V_{l,n} D^*_{l,n} \right) \right) \frac{\partial f^*_l}{\partial t} dx \\
&+ \sum_{l=1}^{L} \int_{R_l} \left(\alpha \Delta f^*_l + \big(1-\alpha\big) \left(\sum_{n=1}^{N} \gamma_n V_{l,n} D^*_{l,n} \right) \right) \frac{\partial \lambda_l}{\partial t} dx
\end{aligned}
\tag{A 39}
$$

In Eq. A 39 the last two integrals over R_l can be set to zero by imposing Eq. 3.6 and Eq. 3.4 respectively. Comparing Eq. A 39 with

$$
\frac{\partial E^*_{shape}}{\partial t} = \int_{C_k} \big(\nabla_{C_k} E^*_{shape} \big) \frac{\partial C_k}{\partial t} \bullet N_k ds_k \tag{A 40}
$$

will yield

$$
\begin{aligned}
&\nabla_{C_k} E^*_{shape} = \\
&\left(\big(1-\alpha\big) \sum_{n=1}^{N} \gamma_n V_{k,n} \left((D^*_{k,n})^2 + \lambda_k D^*_{k,n} - (\overline{D}^*_{k,n})^2 - \overline{\lambda}_{k,n} \overline{D}^*_{k,n} \right) - \nabla \lambda_k \bullet \nabla f_k + \beta_k \kappa_k \right)
\end{aligned}
\tag{A 41}
$$

Using

$$
\frac{\partial C_k}{\partial t} = -\nabla_{C_k} E^*_{shape}
$$

will yeild Eq. 3.5 as the gradient descent PDE for the contour C_k.

Making the substitution $x = u$ in Eq. A 37 we have

$$
\begin{aligned}
E^*_{shape} &= \sum_{l=1}^{L} \beta_l \int_{\partial R_l} ds_l \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} \big(1-\alpha\big)\delta_{l,visible_n} \big(I_n - f^*_l \circ g^{-1}_{l,n}\big)^2 dx \\
&+ \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} \int_{g_{l,n}(R_l)} \big(1-\alpha\big)\delta_{l,visible_n} \big(\lambda_l \circ g^{-1}_{l,n}\big)\big(I_n - f^*_l \circ g^{-1}_{l,n}\big) dx \\
&+ \sum_{l=1}^{L} \int_{R_l} \lambda_l \alpha \Delta f^*_l dx
\end{aligned}
\tag{A 42}
$$

Since the regions as well as the region boundaries of the integrals in the 'double summations' in Eq. A 42 depend on the parameters $p_{l,n,k}, k \in [1\ K]$ of $g_{l,n}$ the derivatives of E^*_{shape} with respect to these parameters will have a region based term and a boundary based term.

For layer l we have

$$\frac{\partial E^*_{shape}}{\partial p_{l,n,k}} = \int_{g_{l,n}(R_l)} -2\gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - f^*_l \circ g^{-1}_{l,n}\big)\big((\nabla f^*_l)\circ g^{-1}_{l,n}\big)\left(\frac{\partial g^{-1}_{ln}}{\partial p_{l,n,k}}\right)dx$$

$$+ \int_{g_{l,n}(R_l)} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big((\nabla\lambda_l)\circ g^{-1}_{l,n}\big)\big(I_n - f^*_l\circ g^{-1}_{l,n}\big) - \big(\lambda_l\circ g^{-1}_{l,n}\big)\big((\nabla f^*_l)\circ g^{-1}_{l,n}\big)\right)\left(\frac{\partial g^{-1}_{ln}}{\partial p_{l,n,k}}\right)dx$$

$$+ \int_{\widehat{C}_{l,n}} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big(I_n - f^*_l\circ g^{-1}_{l,n}\big)^2 - \big(I_n - f^*_{next_n,n}\circ g^{-1}_{next_n}\big)^2\right)\frac{\partial\widehat{C}_{l,n}}{\partial p_{l,n,k}}\bullet\widehat{N}_{l,n}\widehat{ds}_{l,n}$$

$$+ \int_{\widehat{C}_{l,n}} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big(\lambda_l\circ g^{-1}_{l,n}\big)\big(I_n - f^*_l\circ g^{-1}_{l,n}\big) - \big(\lambda_{next_n,n}\circ g^{-1}_{next_n}\big)\big(I_n - f^*_{next_n,n}\circ g^{-1}_{next_n}\big)\right)\frac{\partial\widehat{C}_{l,n}}{\partial p_{l,n,k}}\bullet\widehat{N}_{l,n}\widehat{ds}_{l,n}$$

$$= \int_{g_{l,n}(R_l)} -2\gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\big(I_n - f_l^* \circ g_{l,n}^{-1}\big)\big((\nabla f_l^*) \circ g_{l,n}^{-1}\big)\left(\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}}\right)dx$$

$$+ \int_{g_{l,n}(R_l)} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big((\nabla\lambda_l) \circ g_{l,n}^{-1}\big)\big(I_n - f_l^* \circ g_{l,n}^{-1}\big) - \big(\lambda_l \circ g_{l,n}^{-1}\big)\big((\nabla f_l^*) \circ g_{l,n}^{-1}\big)\right)\left(\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}}\right)dx$$

$$+ \int_{\widehat{C}_{l,n}} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big(I_n - f_l^* \circ g_{l,n}^{-1}\big)^2 - \big(I_n - f_{next_n,n}^* \circ g_{next_n}^{-1}\big)^2\right)\left(\frac{\partial g_{l,n}}{\partial p_{l,n,k}} \circ g_{l,n}^{-1}\right) \bullet \widehat{N}_{l,n}\widehat{ds}_{l,n}$$

$$+ \int_{\widehat{C}_{l,n}} \gamma_n\big(1-\alpha\big)\delta_{l,visible_n}\ldots$$

$$\left(\big(\lambda_l \circ g_{l,n}^{-1}\big)\big(I_n - f_l^* \circ g_{l,n}^{-1}\big) - \big(\lambda_{next_n,n} \circ g_{next_n}^{-1}\big)\big(I_n - f_{next_n,n}^* \circ g_{next_n}^{-1}\big)\right)\left(\frac{\partial g_{l,n}}{\partial p_{l,n,k}} \circ g_{l,n}^{-1}\right) \bullet \widehat{N}_{l,n}\widehat{ds}_{l,n}$$

Making the substitition $u = g_{l,n}(x)$ and using Eq. A 26 we have

$$\frac{\partial E_{shape}^*}{\partial p_{l,n,k}} = \int_{R_l} -2\gamma_n\big(1-\alpha\big)V_{l,n}D_{l,n}^*\big(\nabla f_l^*\big)\left(\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}} \circ g_{l,n}\right)du$$

$$\int_{R_l} \gamma_n\big(1-\alpha\big)V_{l,n}\left(D_{l,n}^*\nabla\lambda_l - \lambda_l\nabla f_l^*\right)\left(\frac{\partial g_{ln}^{-1}}{\partial p_{l,n,k}} \circ g_{l,n}\right)du$$

$$+ \int_{C_l} \gamma_n\big(1-\alpha\big)V_{l,n}\left(\big(D_{l,n}^*\big)^2 - \big(\overline{D}_{l,n}^*\big)^2\right)\left(\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right) \bullet \Big[\text{Jacobian}\,(g_{l,n})\Big]^{-T} N_l ds_l$$

$$+ \int_{C_l} \gamma_n\big(1-\alpha\big)V_{l,n}\big(\lambda_l D_{l,n}^* - \overline{\lambda}_{l,n}\overline{D}_{l,n}^*\big)\left(\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right) \bullet \Big[\text{Jacobian}\,(g_{l,n})\Big]^{-T} N_l ds_l$$

Making the substitution $x = u$ followed by the use of the inverse function theorem Eq. A

25 we have

$$\frac{\partial E^*_{shape}}{\partial p_{l,n,k}} = \int_{R_l} 2\gamma_n\big(1-\alpha\big)V_{l,n}D^*_{l,n}\big(\nabla f^*_l\big)\bullet\left[\text{Jacobian}\,(g_{l,n})\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]dx$$
$$\int_{R_l}\gamma_n\big(1-\alpha\big)V_{l,n}\Big(\lambda_l\nabla f^*_l - D^*_{l,n}\nabla\lambda_l\Big)\bullet\left[\text{Jacobian}\,(g_{l,n})\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]dx$$
$$+\int_{C_l}\gamma_n\big(1-\alpha\big)V_{l,n}\Big(\big(D^*_{l,n}\big)^2-\big(\overline{D}^*_{l,n}\big)^2\Big)\left[\text{Jacobian}\,(g_{l,n})\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]\bullet N_l ds_l$$
$$+\int_{C_l}\gamma_n\big(1-\alpha\big)V_{l,n}\big(\lambda_l D^*_{l,n}-\overline{\lambda}_{l,n}\overline{D}^*_{l,n}\big)\left[\text{Jacobian}\,(g_{l,n})\right]^{-1}\left[\frac{\partial g_{l,n}}{\partial p_{l,n,k}}\right]\bullet N_l ds_l$$

This gives us

$$\begin{aligned}&=\gamma_n\big(1-\alpha\big)\int_{R_l}V_{l,n}G_{l,n,k}\bullet\Big(\big(2D^*_{l,n}+\lambda_l\big)\nabla f^*_l - D^*_{l,n}\nabla\lambda_l\Big)dx\\&+\gamma_n\big(1-\alpha\big)\int_{C_l}V_{l,n}\Big(\big(D^*_{l,n}\big)^2+\lambda_l D^*_{l,n}-\big(\overline{D}^*_{l,n}\big)^2-\overline{\lambda}_{l,n}\overline{D}^*_{l,n}\Big)G_{l,n,k}\bullet N_l ds_l\end{aligned}\tag{3.7}$$

which is Eq. 3.7.

REFERENCES

[1] J. Jackson *et al.*, "Dynamic shape and appearance modelling via moving and deforming layers," *Int. J. of Computer Vision*, vol. 79, pp. 71–84, 2008.

[2] D. Mumford and J. Shah, "Optimal approximation by piecewise smooth functions and associated variational problems," *Comm. Pure Appl. Math.*, vol. 42, pp. 577–685, 1989.

[3] T. Chan and L. Vese, "A level set algorithm for minimizing the mumford-shah functional in image processing," in *IEEE Workshop on Variational and Level Set Methods in Computer Vision*, 2001, pp. 161–168.

[4] J. Guillemaut and A. Hilton, "Space-time joint multi-layer segmentation and depth estimation," in *International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012, pp. 440–447.

[5] D. Sun *et al.*, "Layered segmentation and optical flow estimation over time," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 1768–1775.

[6] L. Torres *et al.*, "A robust motion estimation and segmentation approach to represent moving images with layers," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4, 1997, pp. 2981–2984.

[7] J. Wang and E. Adelson, "Layered representation for motion analysis," in *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, 1993, pp. 361–366.

[8] D. Sun *et al.*, "A fully-connected layered model of foreground and background flow," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 2451–2458.

[9] Y. Li and M. Brown, "Single image layer separation using relative smoothness," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 2752–2759.

[10] Y. Y. *et al.*, "Layered object detection for multi-class segmentation," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 3113–3120.

[11] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.

[12] S. Baker *et al.*, "A layered approach to stereo reconstruction," in *Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 434–441.

[13] Z. Zhu and A. Hanson, "3d lamp: a new layered panoramic representation," in *Proceedings of the IEEE Int. Conf. on Computer Vision*, vol. 2, 2001, pp. 723–730.

[14] V. Kolmogorov *et al.*, "Bi-layer segmentation of binocular stereo video.," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 407–414.

[15] M. Lin and C Tomasi, "Surfaces with occlusions from layered stereo," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1073–1078, 2004.

[16] G. Pryor *et al.*, "Layered active contours for tracking," in *BMVC*, 2007.

[17] Y. Zhou and H. Tao, "A background layer model for object tracking through occlusion," in *Proceedings IEEE Int. Conf. on Computer Vision*, vol. 2, 2003, pp. 1079–1085.

[18] H. Tao *et al.*, "Dynamic layer representation with applications to tracking," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 134–141.

[19] P. Smith *et al.*, "Layered motion segmentation and depth ordering by tracking edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, pp. 479–494, 2004.

[20] F. Pedersini *et al.*, "Combined motion and edge analysis for a layer-based representation of image sequences," in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 1, 1996, pp. 921–924.

[21] A. Criminisi *et al.*, "Bilayer segmentation of live video," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 53–60.

[22] J. Xiao and M. Shah, "Accurate motion layer segmentation and matting," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 698–703.

[23] H. Yalcin *et al.*, "The dense estimation of motion and appearance in layers," in *Conference on Computer Vision and Pattern Recognition Workshop*, 2004, p. 165.

[24] P. Torr *et al.*, "An integrated bayesian approach to layer extraction from image sequences," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 297–303, 2001.

[25] M. Kumar *et al.*, "Learning layered motion segmentations of video," in *IEEE Int. Conf. on Computer Vision*, vol. 1, 2005, pp. 33–40.

[26] D. Sun *et al.*, "Local layering for joint motion estimation and occlusion detection," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 1098–1105.

[27] A. Schodl and I. Essa, "Depth layers from occlusions," in *Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. I–639–I–644.

[28] A. S. and S. H., "Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding," in *Proceedings of IEEE Int. Conf. on Computer Vision*, 1995, pp. 777–784.

[29] Y. Zhang *et al.*, "Motion layer based object removal in videos," in *IEEE Workshops on Application of Computer Vision*, vol. 1, 2005, pp. 516–521.

[30] T. Schoenemann and D. Cremers, "High resolution motion layer decomposition using dual-space graph cuts," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2008, pp. 1–7.

[31] F. Xu and Q. Dai, "Occlusion-aware motion layer extraction under large interframe motions," *IEEE Transactions on Image Processing*, vol. 20, no. 9, pp. 2615–2626, 2011.

[32] L. Zhao *et al.*, "Layered scene models from single hazy images," *IEEE Trans. on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2167 –2179, 2018.

[33] K. Lee *et al.*, "Scene warping: layer-based stereoscopic image resizing," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 49–56.

[34] Q. Ke and T. Kanade, "A subspace approach to layer extraction," in *Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. I–255–I–262.

[35] A. Tsai *et al.*, "A shape-based approach to the segmentation of medical imagery using level sets," *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 137–154, 2003.

[36] V. Borges *et al.*, "Weighted variational two-phase image segmentation based on fuzzy region competition," in *IEEE Int. Conf. on Systems, Man, and Cybernetics*, 2011, pp. 1693–1698.

[37] M. Baust *et al.*, "Translation, scale, and deformation weighted polar active contours," *J. Mathematical Imaging and Vision*, vol. 44, no. 3, pp. 354–365, 2012.

[38] A. Morar *et al.*, "Image segmentation based on active contours without edges," in *IEEE Int. Conf. on Intelligent Computer Communication and Processing*, 2012, pp. 213–220.

[39] M. Aslan *et al.*, "Probabilistic shape-based segmentation method using level sets," *IET Computer Vision*, vol. 8, no. 3, 2014.

[40] T. Khaing and P. Aimmanee, "Optic disk segmentation in retinal images using active contour model based on extended feature projection," in *Int. Conf. of Information and Communication Technology for Embedded Systems*, 2017, pp. 1–6.

[41] A. Khadidos *et al.*, "Patch-based segmentation of overlapping cervical cells using activecontour with local edge information," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2017, pp. 1058–1062.

[42] D. Cremers and S. Soatto, "Variational space-time motion segmentation," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 886–893.

[43] D. Cremers, "A multiphase level set framework for motion segmentation," *Scale-Space 2003, LNCS 2695*, pp. 599–614, 2003.

[44] R. Ciampini *et al.*, "Motion-based segmentation by means of active contours," in *Proceedings Int. Conf. on Image Processing*, vol. 2, 1998, pp. 667–670.

[45] A. Mansouri *et al.*, "Spatio-temporal motion segmentation via level set partial differential equations," in *Proceedings Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, 2002, pp. 243–247.

[46] A. Mansouri and A. Mitiche, "Spatial/joint space-time motion segmentation of image sequences by level set pursuit," in *Proceedings Int. Conf. on Image Processing*, vol. 2, 2002, pp. II–265 –II–268.

[47] A. Mansouri and J. Konrad, "Multiple motion segmentation with level sets," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 201–220, 2003.

[48] O. Besbes and Z. Belhadj, "Multiple motion segmentation with level sets without prior information," in *Image Processing, 2004. ICIP '04*, vol. 1, 2004, pp. 369–372.

[49] G. Sullivan *et al.*, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, 16491668, 2012.

[50] "H.265: high efficiency video coding," ITU-T, Tech. Rep., 2013.

[51] "Objective video quality measurement using a peak-signal-to-noise-ratio (psnr) full reference techniqueh.265: high efficiency video coding," American National Standars Institute, Tech. Rep., 2001.

[52] W. Z. *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 600612, 2004.

[53] C. D. *et al.*, "Vsnr: a wavelet-based visual signal-to-noise ratio for natural images," *IEEE Transactions on Image Processing*, vol. 16, no. 9, 22842298, 2007.